Freie Universität Berlin
Fachbereich Mathematik und Informatik

Institut für Informatik

Masterarbeit

Mixed-Integer Optimization for Loopless Flux Distributions in Metabolic Networks

Hannah Marie Troppens

24. April 2024

Betreuer	Dr. Mathieu Besançon
	Centre Inria d'Université Grenoble Alpes, Zuse Institut Berlin
	Prof. Dr. Sebastian Pokutta
	Technische Universität Berlin, Zuse Institut Berlin
	Dr. St. Elmo Wilken
	Heinrich-Heine-Universität Düsseldorf
Erstgutachter	Prof. Dr. Sebastian Pokutta
Zweitgutachter	Prof. Dr. Ralf Borndörfer
	Freie Universität Berlin, Zuse Institut Berlin

Abstract

Loopless Flux Balance Analysis (ll-FBA) is a constraint-based approach to predict flux distributions in cells that do not contain internal loops, which would be physiologically unrealistic. ll-FBA is a disjunctive program, usually reformulated as a mixed-integer program, and is challenging to solve for biological models that often contain thousands of reactions and metabolites.
In this thesis, we compare various reformulations of ll-FBA and different solution approaches. We discuss the use of intersection cuts and compare the performance of blocking cycles to decomposing the problem and to solving the convex-hull formulation.
Overall, the combinatorial Benders' decomposition is the most promising of the tested approaches with which we could solve most instances. However, the model size and numerical instability pose a challenge to the combinatorial Benders' method.

Zusammenfassung

Loopless Flux Balance Analysis (ll-FBA) ist eine Constraint-basierte Methode zur Modellierung des Stoffwechsels in der Zelle. ll-FBA ist ein disjunktives Programm, das in der Regel als gemischt-ganzzahliges Programm umformuliert wird, und ist für biologische Modelle, die oft Tausende von Reaktionen und Metaboliten enthalten, schwierig zu lösen.
In dieser Arbeit vergleichen wir mehrere Umformulierungen von ll-FBA und unterschiedliche Lösungsstrategien. Wir erörtern die Verwendung von Schnittebenen basierend auf Schnittstellen und testen die Laufzeit, wenn thermodynamisch unmögliche Lösungen durch Ungleichungen von dem Lösungsraum abgeschnitten werden. Außerdem vergleichen wir die Laufzeit, wenn ll-FBA in zwei Teilprobleme zerlegt wird, und wenn wir das Problem so umformulieren, dass wir die konvexe Hülle des disjunktiven Programms erhalten.
Insgesamt ist die kombinatorische Benders-Zerlegung der vielversprechendste der getesteten Ansätze, mit dem wir die meisten Instanzen lösen. Allerdings stellen die Größe des Models und die numerische Instabilität eine Herausforderung für die kombinatorische Benders Methode dar.

Contents

Acronyms

CFF	CycleFreeFlux
FBA	flux balance analysis
ll-FBA	loopless FBA
COBRA	constraint-based reconstruction and analysis
DP	disjunctive program
GEM	genome-scale metabolic model
LP	linear program
MIP	mixed-integer program
MIS	minimal infeasible subsystem
MP	master problem
SP	subproblem

List of Figures

List of Problems

Freie Universität Berlin

Fachbereich Mathematik, Informatik und Physik

SELBSTSTÄNDIGKEITSERKLÄRUNG

Name:	Troppens	(BITTE nur Block- oder Maschinenschrift verwenden.)
Vorname(n):	Hannah Marie	
Studiengang:	Informatik (M.Sc.)	
Matr. Nr.:	5039637	

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum: 22.04.2024 Unterschrift: Hannah Troppens

Introduction

In systems biology, mathematical modeling is used to predict the behavior of biological systems [33]. A fundamental system is the cell and its metabolism. Due to advances in sequencing and gene annotation technology, detailed information about the metabolism of numerous different organisms is available [29]. Genome-scale metabolic networks model the interactions between entities in the cell such as small molecules (metabolites) and enzymes, by making use of these annotated genes [33]. A commonly used method to study metabolic networks is a constraint-based approach, which assumes a steady state of metabolites within the cell, ensuring a mass balance between the production and the consumption of each metabolite [29]. The steady-state assumption, together with flux bounds on each reaction, defines a system of linear constraints. A solution corresponds to a flux distribution, which is the rate of each reaction, that satisfies the constraints [33].
A basic method to analyze the flux distribution in a metabolic network is flux balance analysis (FBA) [28]. In FBA, a linear program is solved where we optimize a biological objective function, such as maximizing growth. As cells evolve under selection pressure, it is plausible that the behavior is in line with optimizing growth and using mathematical optimization to study the flow of metabolites through a metabolic network is reasonable [29]. With FBA, it is possible to predict cellular behavior in accordance with observations in experiments [28].
However, solutions of FBA often violate the loop law, which means that the solution contains internal cycles which is biologically implausible. Loopless FBA (ll-FBA) is an extension to FBA which includes thermodynamic information [37]. A solution to ll-FBA does not contain internal cycles, but ll-FBA is computationally harder as we need to solve a disjunctive program, which is usually reformulated as a mixed-integer program (MIP). In this thesis, we look at different reformulations of ll-FBA and compare the performance of different solution approaches.

In Chapter 1, we will give a short introduction to mathematical optimization with a focus on the aspects relevant to this thesis including the problem definitions and solving strategies for linear programming, mixed-integer programming and disjunctive

programming. We will present the biological context in more detail and present the relevant problems in detail in Chapter 2. We consider several reformulations and solution strategies in Chapter 3, including a combinatorial Benders' approach, and we establish how to apply intersection cuts to ll-FBA. In Chapter 4, we show and discuss the computational results.

Preliminaries and Notation

1. A vector $\mathbf{v} = (v_1, ..., v_n) \in \mathbb{R}^n$ is identified as a column vector and printed in bold. The transpose of $\mathbf{v}$ into a row vector is written as $\mathbf{v}^\intercal$. The inner product of two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ is $\mathbf{v}^\intercal\mathbf{w} = \sum_{i=1}^{n} v_i w_i$. With $\mathbf{v} \leq \mathbf{w}$ we denote element-wise inequality. With $\ln(\mathbf{v})$ we denote the element-wise natural logarithm. The 1-vector is written as $\mathbf{1} := (1, 1, ..., 1) \in \mathbb{R}^n$ the 0-vector as $\mathbf{0} := (0, 0, ..., 0) \in \mathbb{R}^n$. The *support* of $\mathbf{v}$ is the set of indices i with $v_i \neq 0$ and is denoted by $\text{supp}(\mathbf{v})$. With $\text{sign}(\mathbf{v})$ the element-wise sign function is applied to $\mathbf{v}$. The concatenation of two vectors $\mathbf{x}$, $\mathbf{y}$ is denoted by $(\mathbf{x}, \mathbf{y})$.

2. The entry in row i and column j of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is denoted by $a_{i,j}$. The i-th row is $\boldsymbol{a}_{i,*}$ and the j-th column $\boldsymbol{a}_{*,j}$. The zero matrix is denoted by $\mathbf{0}_{m,n}$ with m rows and n columns. With $\text{diag}(\mathbf{v})$ we denote the quadratic matrix with $\mathbf{v}$ on the diagonal and 0 for all other entries.

3. The *rank* of the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ corresponds to the number of linearly independent columns of $\mathbf{A}$ and is denoted by $\text{rank}(\mathbf{A})$. The matrix $\mathbf{A}$ is said to be of *full rank* if $\text{rank}(\mathbf{A}) = \min\{m, n\}$. The number of linearly independent rows and linearly independent columns is always equal.

4. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an invertible matrix: $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The matrix $\mathbf{A}^{-1}$ is the inverse matrix of $\mathbf{A}$. $\mathbf{I}_n$ is the $n \times n$ identity matrix.

5. A *linear combination* is defined as $\sum_{i=1}^{m} \lambda_i \mathbf{x}_i = \lambda_i x_1 + ... + \lambda_m x_m$, where $\lambda_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$. A line going through a point $\mathbf{x}$ generated by $\mathbf{r} \in \mathbb{R}^n$ is the set $\{\mathbf{x} + \lambda\mathbf{r} | \lambda \in \mathbb{R}\}$. A *line segment* is a subset of a line defined on the interval between $l \in \mathbb{R}$ and $u \in \mathbb{R}$: $\{\mathbf{x} + \lambda\mathbf{r} | \lambda \in \mathbb{R},\ \lambda \in [l, u]\}$.

6. A *basis* B of a vector space V is a set of vectors $(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n)$ that are linearly independent and every $\mathbf{v} \in V$ can be written as a linear combination of vectors in B.

7. The *nullspace* of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as $\text{null}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$.

8. A set $C \subseteq \mathbb{R}^n$ is *convex* if for any two points $\mathbf{x}, \mathbf{y} \in C$ the line segment between $\mathbf{x}$ and $\mathbf{y}$ is in C. The *convex hull* of a set X is the smallest convex set that contains all points in X and is denoted by $\text{conv}(X)$. It is a set of convex combinations such that all points x_i in X can be represented, where a *convex combination* is a linear combination with $\lambda_i \geq 0$ and $\sum_{i=1}^{m} \lambda_i = 1$.

9. The set $\{\mathbf{x} \in \mathbb{R}^n | \boldsymbol{\alpha}^\intercal \mathbf{x} = \boldsymbol{\beta}\}$ is a *hyperplane*, where $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{\beta} \in \mathbb{R}$. The set $\{\mathbf{x} \in \mathbb{R}^n | \boldsymbol{\alpha}^\intercal x \leq \boldsymbol{\beta}\}$ is a *half-space*. A *polyhedron* $P = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is the intersection of a finite number of half-spaces, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Hyperplanes, half-spaces and polyhedra are convex. A point $\mathbf{x} \in P$ is an *extreme point* or *vertex* if it cannot be represented as a convex combination of any set of other points in P. A *polytope* is a bounded polyhedron if there exists a large enough ball in which it can be placed [40].

10. A set $C \subseteq \mathbb{R}^n$ is a *cone* if $\lambda \mathbf{x} \in C$ for any $\mathbf{x} \in C$ and $\lambda \geq 0$. A *conic combination* is a linear combination with $\lambda_i \geq 0$. C is *pointed* if it contains no line and the extreme point is called *apex*. A nonzero vector $\mathbf{r} \in \mathbb{R}^n$ is a *ray* of C if $\{\mathbf{x} + \lambda \mathbf{r} | \lambda \geq 0\} \in C$ for any $\mathbf{x} \in C$. Ray $\mathbf{r}$ is an *extreme ray* if it cannot be represented by a conic combination of other rays in C. A cone is *polyhedral* if the number of extreme rays is finite. A cone is *simplicial* if it has n extreme rays [5]. The *conic relaxation* of an extreme point $\mathbf{x}$ of a polyhedron P is a cone with apex $\mathbf{x}$, and the extreme rays are the half-spaces of P intersecting at $\mathbf{x}$.

11. A set $S \subseteq \mathbb{R}^n$ is *open* if for any point $\mathbf{x} \in S$ there exists an $\epsilon > 0$ such that the ball centered at $\mathbf{x}$ with radius ϵ is contained in S. A set is *closed* if its complement is open. In particular, the set of whole numbers $\mathbb{Z} \subseteq \mathbb{R}$ is closed.

Chapter 1

Mathematical Optimization

In mathematical optimization, the goal is to find an optimal solution that respects specific requirements. In particular, the behavior in cells can be modeled by mathematical optimization with biological objectives such as maximizing growth or minimizing energy usage [33]. Optimization in cells is the topic of this thesis, and we will see in Chapter 2 how to use mathematical optimization to answer questions arising in systems biology.
In order to express a problem in practice mathematically, we need to formulate an objective and identify the constraints, i.e. the requirements in our system. The resulting model is an *optimization problem*, and we can use optimization algorithms to compute solutions. We define an optimization problem as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}) \tag{1a}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \tag{1b}$$

where f is the *objective function*, g_i are the *constraint functions* and $\mathbf{x}$ are the *decision variables*. The *feasible region* is the set of points that respect the constraints. An optimization problem with a constrained feasible region is often also called *constrained optimization problem* in contrast to an *unconstrained optimization problem*. A *solution* is a vector $\mathbf{x}$ that lies in the feasible region. An *optimal solution* $\mathbf{x}^*$ is a solution with the smallest objective value. The *objective value* of $\mathbf{x}^*$ is the value of the objective function evaluated at $\mathbf{x}^*$. If a problem has no solution, meaning that the constraints cannot be satisfied simultaneously, it is said to be *infeasible*. A problem is *unbounded* if solutions exist, but the objective value can be arbitrarily small.

One can maximize a function f by setting the objective function to $\min -f(\mathbf{x})$. Depending on the type of objective function and the types of constraints, optimization problems are divided into different classes. The classes relevant for this thesis are linear programs, mixed-integer programs and disjunctive programs[1]. Different algorithms can be used to solve optimization problems, which depend on the problem structure.

1.1 Linear Programming

A *linear program* (LP) is an optimization problem with a linear objective function and linear constraints.

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{2a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{2b}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{2c}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. It is assumed that the number of constraints m is greater or equal to the number of variables n. We also assume that the columns of $\mathbf{A}$ are linearly independent.The linear inequalities define a polyhedron. If an optimal solution exists and if the feasible region has vertices, there exists an optimal solution at one of the vertices. In that case, the LP can have exactly one optimal solution or multiple optimal solutions, for example, if an entire edge or face is optimal. An LP has no optimal solution if it is infeasible or unbounded. Problem (2) is said to be in *inequality form* [6].

As an example, we want to solve the following optimization problem:

$$\max_{x,y} \quad y \tag{3a}$$

$$\text{s.t.} \quad 0 \leq x \leq 3 \tag{3b}$$

$$0.5 \leq y \leq 4 \tag{3c}$$

$$y \leq 1.5x + 0.5 \tag{3d}$$

$$y \leq -0.5x + 4.5 \tag{3e}$$

$$x, y \in \mathbb{R} \tag{3f}$$

The decision variables x, y are continuous. We want to find a solution (x^*, y^*) with maximal y-value such that the constraints are respected. The problem is a linear

[1] Program in this context does not mean a computer program. The term was coined in the 1950s and referred to planning in a military context. See [40] for more detail.

program and can be written as Problem (2) by simple linear transformations. If we want to write Problem (3) as a minimization problem, the objective becomes $-y$. If we visualize the example (Figure 1), we see that there exists one optimal solution located at $(2, 3.5)$. Usually, we are interested in problems in higher dimensions, and it is not possible to solve them visually.

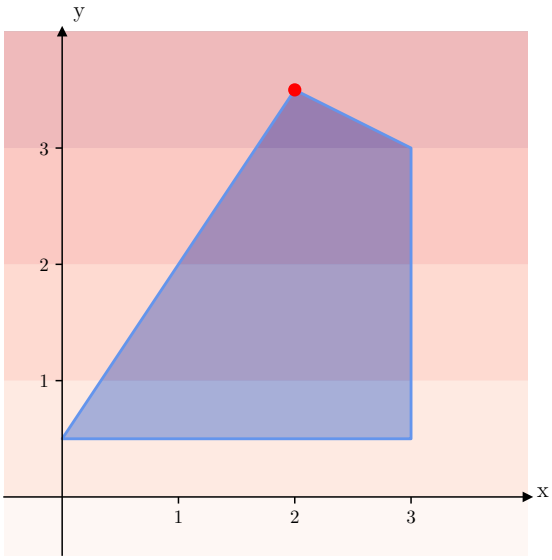


Figure 1: Visualization of LP (Problem (3))

The set of feasible solutions contains all the points in the interior or on the boundary of the polytope (blue area). The optimal solution is the point in the feasible region with the biggest y-value (red point). The isolines indicate the function value of the points.

An LP is said to be in *standard form* if it is of the form:

$$\min_{x} \quad \mathbf{c}^\intercal \mathbf{x} \tag{4a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{4b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{4c}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Problem (2) can be written in standard form by adding one *slack variable* per inequality to write it as equality [27]. The constraint $\boldsymbol{a}_{i,*}^\intercal \mathbf{x} \leq b_i$ becomes $\boldsymbol{a}_{i,*}{}^\intercal \mathbf{x} + s_i = b_i$ with $s_i \geq 0$. Each variable x_i that can be negative is replaced by $x_i^+ - x_i^-$, where $x_i^+ \geq 0$ and $x_i^- \geq 0$ [27].
A vertex of the feasible region is also called *basic feasible solution*. Suppose we have an LP in standard form as in Problem (4) with linearly independent rows. Let $\mathbf{x} \in \mathbb{R}^n$ be a basic feasible solution. There exist m constraints that are a basis defining $\mathbf{x}$. B is the set of constraint indices that are in the basis. Any variable $x_i \in B$ is a *basic variable*. A variable $x_i \notin B$ is a *nonbasic variable* and $x_i = 0$ [40].

1.1.1 Solving LPs

There exist several algorithms to solve linear programs. The *ellipsoid method* is the first algorithm that was proven to have a polynomial running time in the worst case. However, in practice, other algorithms outperform it [40]. Another class of algorithms is *interior-point methods*. Some interior-point methods have a polynomial running time in the worst case and are often used in practice. Interior-point methods start with a feasible solution in the interior of the feasible region and approach the optimum without stepping outside the feasible region [40].
Another algorithm that is relevant in practice is the *simplex algorithm*. It is based on the fundamental property of LPs that the optimum is located at one of the vertices of the feasible region. In *phase I* of the algorithm, a vertex of the feasible region is computed. In *phase II*, the algorithm moves from vertex to vertex along the edges of the feasible region until an optimum is found. The *pivot rule* determines to which vertex the algorithm moves next. There exist several pivot rules, however, for all of them, there are families of problem instances on which the number of pivot steps needed grows exponentially. The worst-case running time on some instances conflicts with the observed polynomial running time in practice. Studying the simplex method with *smoothed analysis*, which tries to close this gap, shows a polynomial running time of the simplex method. In smoothed analysis, a small noise is added to the entries of a fixed instance and afterward, worst-case analysis is performed on the perturbed instance [17, 11].

1.1.2 Optimality and Duality

To prove that a solution is optimal, we can use the *Karush-Kuhn-Tucker*-conditions (KKT-conditions) and see the relation between the primal and the dual problems in LPs. An optimization problem can be written as an unconstrained problem by augmenting the objective function with a weighted sum of the constraints [6]. The resulting function is known as the *Lagrangian* function. The Lagrangian of an LP in standard form is [27]:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{c}^\intercal \mathbf{x} - \sum_{i=1}^{n} \lambda_i x_i - \sum_{i=1}^{m} \nu_i (A_{i,*}^\intercal \mathbf{x} - b_i) \tag{5}$$

where $\lambda_i \geq 0$ and $\nu_i \in \mathbb{R}$ are called *Lagrange multipliers* or *dual variables*. As the objective function of linear programs is convex, the KKT-conditions are a proof for global optimality of a solution. We obtain the optimality conditions captured in Theorem 1.

Theorem 1 (Optimality conditions for LPs) *A solution* $\mathbf{x}^*$ *is optimal if there exist vectors* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\nu}$ *that satisfy the following conditions [27]:*

1. $\boldsymbol{\lambda} + \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{c}$ *(stationarity)*
2. $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$ *(primal feasibility)*
3. $\mathbf{x} \geq \mathbf{0}$ *(primal feasibility)*
4. $\boldsymbol{\lambda} \geq \mathbf{0}$ *(dual feasibility)*
5. $\mathbf{x}^\intercal \boldsymbol{\lambda} = \mathbf{0}$ *(complementary slackness)*

We call a linear program in standard form as in Problem (4) the *primal problem* ($\mathcal{P}$). The associated *dual function* of the primal problem is [22, 6]:

$$\begin{aligned} q(\boldsymbol{\lambda}, \boldsymbol{\nu}) &= \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &= \inf_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x} - \boldsymbol{\lambda}^\intercal \mathbf{x} - \boldsymbol{\nu}^\intercal (\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= \inf_{\mathbf{x}} \mathbf{c}^\intercal \mathbf{x} - \boldsymbol{\lambda}^\intercal \mathbf{x} - \boldsymbol{\nu}^\intercal \mathbf{A}\mathbf{x} + \boldsymbol{\nu}^\intercal \mathbf{b} \\ &= \inf_{\mathbf{x}} (\mathbf{c}^\intercal - \boldsymbol{\lambda}^\intercal - \boldsymbol{\nu}^\intercal \mathbf{A})\mathbf{x} + \boldsymbol{\nu}^\intercal \mathbf{b} \\ &= \inf_{\mathbf{x}} \mathbf{x}^\intercal (\mathbf{c} - \boldsymbol{\lambda} - \mathbf{A}^\intercal \boldsymbol{\nu}) + \mathbf{b}^\intercal \boldsymbol{\nu} \\ &= \begin{cases} \mathbf{b}^\intercal \boldsymbol{\nu} & \text{if } \mathbf{c} - \boldsymbol{\lambda} - \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{0} \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

Let $\mathbf{x}^P$ be a feasible solution to the primal problem, and let $(\boldsymbol{\lambda}^D, \boldsymbol{\nu}^D)$ be a feasible solution to the dual problem. If the dual function is bounded we know that $\mathbf{c} - \boldsymbol{\lambda}^D = \mathbf{A}^\intercal \boldsymbol{\nu}^D$ and we know that $\mathbf{x}^P$ satisfies $\mathbf{A}\mathbf{x}^P = \mathbf{b}$. The function value $q(\boldsymbol{\lambda}^D, \boldsymbol{\nu}^D)$ is a lower bound on the objective value of the primal problem [22]:

$$\mathbf{b}^\intercal \boldsymbol{\nu}^D = (\mathbf{A}\mathbf{x}^P)^\intercal \boldsymbol{\nu}^D = \mathbf{x}^{P\intercal} \mathbf{A}^\intercal \boldsymbol{\nu}^D = \mathbf{x}^{P\intercal} (\mathbf{c} - \boldsymbol{\lambda}^D) = \mathbf{c}^\intercal \mathbf{x}^P - \boldsymbol{\lambda}^{D\intercal} \mathbf{x}^P \leq \mathbf{c}^\intercal \mathbf{x}^P$$

The tightest bound maximizes $\mathbf{b}^\intercal \boldsymbol{\nu}$. Formulated as a linear program we obtain the *dual problem* ($\mathcal{D}$):

$$\begin{aligned} \max_{\boldsymbol{\nu}, \boldsymbol{\lambda}} \quad & \mathbf{b}^\intercal \boldsymbol{\nu} && \text{(6a)} \\ \text{s.t.} \quad & \boldsymbol{\lambda} + \mathbf{A}^\intercal \boldsymbol{\nu} = \mathbf{c} && \text{(6b)} \\ & \boldsymbol{\lambda} \geq \mathbf{0} && \text{(6c)} \end{aligned}$$

or rewritten without the slack variables $\boldsymbol{\lambda}$ [27]:

$$\max_{\boldsymbol{\nu}} \quad \mathbf{b}^\intercal \boldsymbol{\nu} \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{A}^\intercal \boldsymbol{\nu} \leq \mathbf{c} \tag{7b}$$

Let p^* be the objective value of an optimal solution to the primal problem and d^* the objective value of an optimal solution to the dual problem. We have seen that d^* is a lower bound on p^* which is known as *weak duality*. We speak of *strong duality* if $p^* = d^*$.

Theorem 2 (Strong Duality) *Given a primal $\mathcal{P}$ and corresponding dual program $\mathcal{D}$, exactly one of the following is true [27]:*

1. *$\mathcal{P}$ and $\mathcal{D}$ both have at least one optimal solution. If p^* is the objective value of an optimal solution to $\mathcal{P}$ and d^* is the objective value of an optimal solution to $\mathcal{D}$, then $p^* = d^*$.*
2. *Either $\mathcal{P}$ or $\mathcal{D}$ is unbounded and the other is infeasible.*
3. *$\mathcal{P}$ and $\mathcal{D}$ both are infeasible.*

A proof for Theorem 2 can be found in [22]. Property (1) of Theorem 2 can be used to prove the optimality of a solution.

We have seen how to derive the dual problem of an LP in standard form. We do not require an LP in standard form to derive the corresponding dual problem. For example, the dual problem of Problem (2) is:

$$\max_{\boldsymbol{\mu}} \quad \mathbf{b}^\intercal \boldsymbol{\mu} \tag{8a}$$

$$\text{s.t.} \quad \mathbf{A}^\intercal \boldsymbol{\mu} = \mathbf{c} \tag{8b}$$

$$\mu_i \geq 0 \tag{8c}$$

1.2 Mixed-Integer Programming

Many problems in practice cannot be formulated by only using linear constraints and continuous decision variables. It is often required that a subset of variables is discrete. A *mixed-integer program* (MIP) is an optimization problem with a linear objective function, linear constraints and a subset of integer variables:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{9a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{9b}$$

$$\mathbf{x} \in \mathbb{Z}^{|J|} \times \mathbb{R}^{n-|J|} \tag{9c}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. The set J contains the indices of integer variables.

Let us reuse the LP example in Section 1.1 and add integrality constraints on the decision variables x and y:

$$\max_{x,y} \quad y \tag{10a}$$

$$\text{s.t.} \quad 0 \leq x \leq 3 \tag{10b}$$

$$0.5 \leq y \leq 4 \tag{10c}$$

$$y \leq 1.5x + 0.5 \tag{10d}$$

$$y \leq -0.5x + 4.5 \tag{10e}$$

$$x, y \in \mathbb{Z} \tag{10f}$$

Looking at the visualization of the problem (Figure 2), we see that the optimal solution of the LP $x^{LP} = (2, 3.5)$ is no valid solution to the MIP formulation. The points $(2, 3)$ and $(3, 3)$ are the optimal solutions.

The MIP formulation enables us to model much more complex problems. Apart from incorporating discrete quantities, it is possible to capture Boolean expressions. Suppose we want to model the *indicator constraint* $z = 1 \implies \boldsymbol{a}^\intercal \mathbf{x} \leq b$. We can reformulate the constraint with a linear constraint using the *big-M method*:

$$\boldsymbol{a}^\intercal \mathbf{x} \leq b + M(1 - z)$$

If $z = 1$ the constraint is enforced. In the case $z = 0$, the value of M has to be larger than $\boldsymbol{a}^\intercal \mathbf{x} - b$ for any $\mathbf{x}$ such that the constraint is inactive [22].

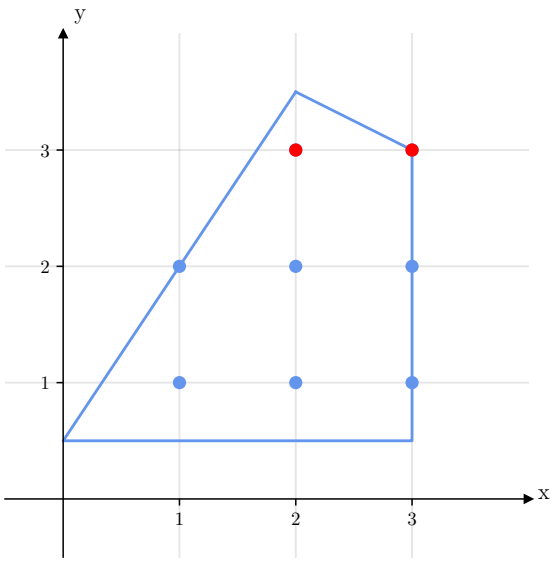


Figure 2: Visualization of MIP (Problem (10))

The feasible points of the MIP are the integer points respecting the polyhedral constraints (blue points). The set of feasible solutions to the relaxed LP are all the points in the interior or on the boundary of the polytope (blue line segments). An optimal solution is a point in the feasible region with the biggest y-value (red point). The optimal solutions are at $(2, 3)$ and $(3, 3)$.

Solving MIPs

Solving MIPs is much more complicated than solving LPs, because it is not guaranteed that if an optimal solution exists, it is at one of the vertices. In general, solving MIPs is $\mathcal{NP}$-hard. A problem is in $\mathcal{NP}$ if a solution is verifiable in polynomial time. A problem is $\mathcal{NP}$-hard if it requires at least as much time to solve the problem as any other problem that is in $\mathcal{NP}$. $\mathcal{NP}$-complete problems are $\mathcal{NP}$-hard and are in $\mathcal{NP}$ [10]. In complexity theory, problems are usually defined as decision problems. The decision problem of Problem (9) would be "Is the feasible region nonempty?", which is $\mathcal{NP}$-complete [8]. As finding an optimal solution is not easier than finding any solution, it follows that MIPs are $\mathcal{NP}$-hard [8].

Instead of solving the MIP directly, one can solve a sequence of *LP relaxations*: the integrality constraints are ignored and Constraint (9c) becomes $\mathbf{x} \in \mathbb{R}^n$. Let z^{LP} be the objective value of an optimal solution $\mathbf{x}^{LP}$ of the LP relaxation and z^* the objective value of an optimal solution $\mathbf{x}^*$ to the MIP problem.
We know that:

$$z^{LP} \leq z^*$$

One common approach for solving MIPs is the *branch-and-bound* algorithm [8]. The idea is to generate a branch-and-bound tree starting with the solution to the LP relaxation $\mathbf{x}^{LP} \in \mathbb{R}^n$ at the root node. A variable x_i that is fractional in $\mathbf{x}^{LP}$ and violates the integrality constraint is selected as *branching variable*. We divide the search space P by creating two child nodes. In one x_i has to be larger or equal to the ceiled value and the feasible region becomes $P \cap \{x_i \geq \lceil x_i^{LP} \rceil\}$. In the other child node, x_i can take at most the floored value of x_i^{LP} and the feasible region of the subproblem is $P \cap \{x_i \leq \lfloor x_i^{LP} \rfloor\}$. The solution with the smallest objective value respecting the MIP formulation is called *incumbent* and the objective value is denoted by z^{INC}. We continue branching until all variables in J are integral, a subproblem is infeasible or a node can be *pruned*. The optimal solution in each node i is bounded by the objective value of the relaxed solution $z_{(i)}^{LP}$. If $z_{(i)}^{LP}$ is greater or equal to z^{INC}, node i is cut off the tree.

Another approach to solve MIPs is the *cutting plane* algorithm [8]. The LP relaxation can be very weak. As we are dealing with a linear objective, we could get the optimal MIP solution easily if we had access to the *integer hull*: $\text{conv}(P \cap \mathbb{Z}^n)$. If the LP relaxation does not correspond to the integer hull, one can tighten the LP relaxation by adding *cuts*. A cut is an inequality that does not cut off any feasible solution (see Section 1.4). Given a relaxed solution $\mathbf{x}^{LP}$ that violates at least one integer constraint, one separates $\mathbf{x}^{LP}$ from the hull with a cut. This process is repeated until $\mathbf{x}^{LP}$ is an optimal solution to the MIP.

A combination of the branch-and-bound algorithm and the cutting plane algorithm is the *branch-and-cut* algorithm [8]. The LP relaxation at a node is potentially tightened by adding cuts.

1.3 Disjunctive Programming

Often, the requirements of a problem are complex, and linear constraints are not sufficient for modeling. With Boolean expressions, more complicated relationships between variables can be captured. A *disjunctive program* (DP) is an optimization problem with linear constraints, continuous variables and logical constraints:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{11a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{11b}$$

$$\bigvee_{i \in Q_j} \{\mathbf{d}^{(i)\intercal}\mathbf{x} \leq d_0^i\} \quad \forall j \in S \tag{11c}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{d}^{(i)} \in \mathbb{R}^n$ and $d_0^i \in \mathbb{R}$. The set S is the set of disjunction indices. As the terms i in each disjunction Q_j are linear, each disjunctive set is a polyhedron. The feasible region is in general non-convex due to the disjunctive constraints [2]. Alternatively, the disjunctions can be captured by Boolean variables $Y_{ij} \in \{true, false\}$, where Y_{ij} corresponds to the i-th disjunct in disjunction j:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{12a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{12b}$$

$$\bigvee_{i \in Q_j} \begin{bmatrix} Y_{ij} \\ \{\mathbf{d}^{(i)\intercal}\mathbf{x} \leq d_0^i\} \end{bmatrix} \quad \forall j \in S \tag{12c}$$

$$\Omega(Y) = true \tag{12d}$$

As an example, we want to solve the following optimization problem:

$$\max_{x,y} \quad y \tag{13a}$$

$$\text{s.t.} \quad \Big((0 \leq x \leq 1) \wedge (0.5 \leq y \leq 1.5x + 0.5)\Big) \vee \Big((2 \leq x \leq 3) \wedge (2 \leq y \leq -0.5x + 4.5)\Big) \tag{13b}$$

$$x, y \in \mathbb{R} \tag{13c}$$

The feasible region is the union of two polyhedra and no longer convex. The optimal solution is the point with maximal value in either of the polytopes and located at $(2, 3.5)$ which we see in the visualization (Figure 3).

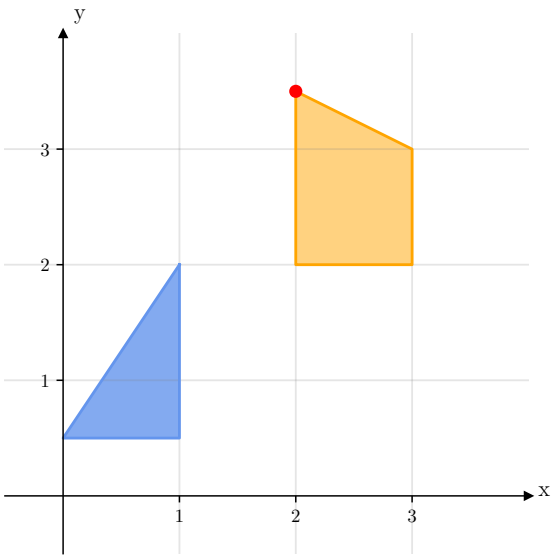


Figure 3: Visualization of DP (Problem (13))

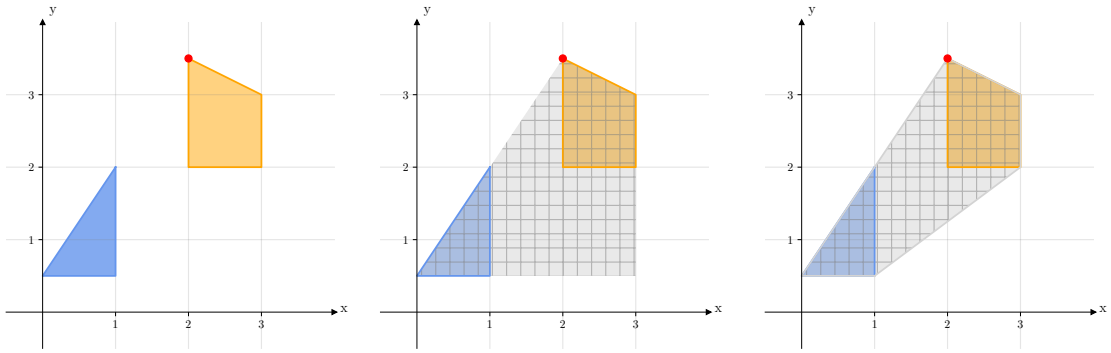
The set of feasible solutions is no longer convex and is the union of the polytope P_1 (blue area) and the polytope P_2 (orange area). The optimal solution is the point in the feasible region with the biggest y-value (red point).

Solving DPs

A disjunctive model can be formulated as a mixed-integer program and solved by corresponding techniques (see Section 1.2). Disjunctions can be expressed by linear constraints and integer variables by using the big-M method.

Another possibility is to convexify the feasible region and solve the resulting linear program. The idea is to build the convex hull of the union of polyhedral points in a higher dimension. The optimal solution at a vertex of the convex hull is optimal for the disjunctive program, as we are dealing with linear constraints. For an example of how to write a disjunctive program using the big-M or the hull reformulation, we refer to [31]. Figure 4 shows the 2D projection of the big-M reformulation and the convex-hull formulation of a disjunctive program.

Figure 4: DP reformulations



(a) (b) (c)

The feasible region of a disjunctive program (a), the approximation of the region using the big-M reformulation (b) and the convex-hull formulation (c). The plot is inspired by [1].

1.4 Decompositions and Cutting Planes

Let us consider a mixed-integer problem (Problem (9)) with decision variables $\mathbf{x}$. A *cutting plane* or *cut*, parameterized by $\boldsymbol{\alpha} \in \mathbb{R}^{n+1}$, is an inequality that when added to the MIP does not remove any feasible solution and is defined as:

$$\sum_{i=1}^{n} \alpha_i x_i \leq \alpha_0 \quad \forall \mathbf{x} \in P \cap S \tag{14}$$

where $\alpha_i \in \mathbb{R}$, P is the polyhedron defined by the inequality in Constraint (9b), and S is the set of points satisfying Constraint (9c). Usually, cutting planes are used to cut off solutions to the relaxed problem that are infeasible in the original problem.

Figure 5 shows a visualization of Problem (10) in which the relaxed solution $\mathbf{x}^{LP}$ is not feasible in the MIP. The relaxed solution $\mathbf{x}^{LP}$ is separated from the integer hull by cuts.

For a relaxed solution $\mathbf{x}^{LP}$ with $\mathbf{x}^{LP} \notin P \cap S$, there exist multiple hyperplanes separating $\mathbf{x}^{LP}$ from the actual feasible region. As we want to tighten the search space, we are interested in cuts that cut off many infeasible points at once. There are different scores to estimate the quality of a cut [39].

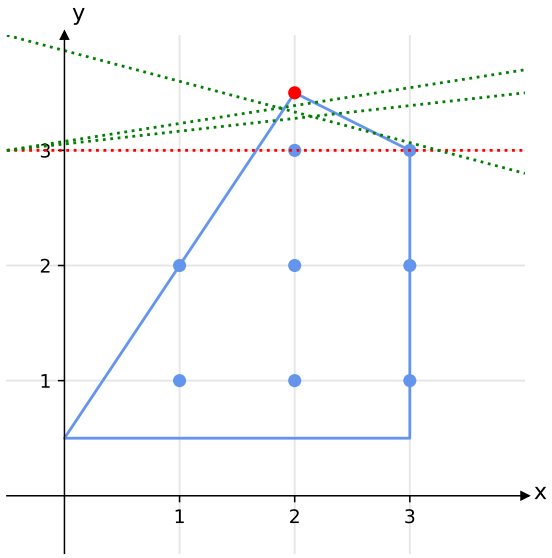


Figure 5: Tightening a relaxed problem with cuts (Problem (10))

The feasible points of Problem (10) are the integer points within the polytope. In the relaxed problem, the integrality constraints are ignored. The relaxed solution is $\mathbf{x}^{LP} = (2, 3.5)$. As $\mathbf{x}^{LP}$ is not an integer solution, it is cut off by adding cuts. The tightest cut is indicated by the red dashed line.

One scoring measure is *efficacy* which is the signed distance from $\mathbf{x}^{LP}$ to the cutting plane:

$$\text{eff}(\boldsymbol{\alpha}, \mathbf{x}^{LP}) := \frac{\boldsymbol{\alpha}^\intercal \mathbf{x}^{LP} - \alpha_0}{\|\boldsymbol{\alpha}\|} \tag{15}$$

However, there is a tradeoff between the quality of a cut and the complexity of generating it.

1.4.1 Decomposition with No-Good Cuts

Given an integer problem P as in Problem (9) with only binary variables: $|J| = n$ and $x_i \in \{0, 1\}$. We can decompose P into a relaxed problem and solve a subproblem to check whether a relaxed solution lies in the feasible region of the original problem. If a relaxed solution is not feasible in the original problem, we add a cut to the relaxed problem to remove the relaxed solution from the search space.
Let $\mathbf{x}^{IP}$ a solution to the relaxed integer program, where Constraint (9b) is relaxed. We further assume that $\mathbf{x}^{IP}$ is not a feasible solution to P. The following *no-good cut* is added to the relaxed IP and forbids the assignment of integer variables in $\mathbf{x}^{IP}$:

$$\sum_{j \in J : x_j^{IP} = 0} (1 - x_j) + \sum_{j \in J : x_j^{IP} = 1} x_j \quad \leq \quad |J| - 1$$

Such a cut usually tightens the feasible region of P marginally and it would require a large number of no-good cuts to arrive at the integer hull.

1.4.2 Combinatorial Benders' Decomposition

Instead of forbidding one assignment of variables as with a no-good cut, with combinatorial Benders' cuts, we generate stronger cuts, by identifying the subset of variables that lead to the infeasibility. This section is based on [7].

Given a problem of the form:

$$\min_{\mathbf{x},\mathbf{z}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{16a}$$

$$\text{s.t.} \quad \mathbf{F}\mathbf{x} \leq \mathbf{g} \tag{16b}$$

$$\mathbf{D}\mathbf{z} \leq \mathbf{e} \tag{16c}$$

$$x_j = 1 \implies \boldsymbol{a}_{i,*}^\intercal \mathbf{z} \leq b_i \quad \forall i \in I \tag{16d}$$

$$x \in \{0,1\}^n \tag{16e}$$

$$z_i \in \mathbb{R} \tag{16f}$$

As the objective does not depend on $\mathbf{z}$ and as the decision variables $\mathbf{x}, \mathbf{z}$ are independent apart from the indicator constraints, we can split the problem into a *master problem* (MP) and a *subproblem* (SP). Constraint (16d) can be reformulated using the big-M method: $\mathbf{A}\mathbf{z} \leq \mathbf{b} + M(\mathbf{1} - \mathbf{x})$.

In the master problem, the constraints on $\mathbf{z}$ are ignored:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{17a}$$

$$\text{s.t.} \quad \mathbf{F}\mathbf{x} \leq \mathbf{g} \tag{17b}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{17c}$$

$$x_j \in \{0,1\} \tag{17d}$$

The subproblem depends on a solution $\mathbf{x}^{MP}$ to the master problem.

$$\min_{\mathbf{z}} \quad 0 \tag{18a}$$

$$\text{s.t.} \quad \mathbf{D}\mathbf{z} \leq \mathbf{e} \tag{18b}$$

$$\boldsymbol{a}_{i,*}^\intercal \mathbf{z} \leq b_i + M(1 - x_j^{MP}) \quad \forall i \in I \tag{18c}$$

If solution $\mathbf{x}^{MP}$ leads to a feasible subproblem, $\mathbf{x}^{MP}$ is an optimal solution to Problem (16). If the problem is infeasible, $\mathbf{x}^{MP}$ is not a feasible solution to the original problem in which case we want to add a cut that removes $\mathbf{x}^{MP}$ from the search space. Suppose we have access to a *minimal infeasible subsystem* (MIS) C that is an inclusion-wise minimal set of row indices corresponding to the constraints in the subproblem that lead to infeasibility.
The subproblem can then be written as:

$$\min_{\mathbf{z}} \quad 0 \tag{19a}$$

$$\text{s.t.} \quad \mathbf{Dz} \leq \mathbf{e} \tag{19b}$$

$$\boldsymbol{a}_{i,*}^{\intercal}\mathbf{z} \leq b_i + M_i(1 - x_j^{MP}) \quad \forall i \in C \tag{19c}$$

The *combinatorial Benders' cut* is then:

$$\sum_{j \in C : x_j^{MP} = 0} (1 - x_j) + \sum_{j \in C : x_j^{MP} = 1} x_j \quad \leq \quad |C| - 1$$

A minimal infeasible subsystem can be found by studying the dual problem of the infeasible subproblem. As the subproblem is a linear problem, strong duality holds and an infeasible primal problem implies an unbounded dual problem (see Section 1.1).

To derive the dual problem of Problem (18) we write constraints (b) and (c) as one constraint and stack the variables in the vector $\mathbf{y}$: $\tilde{\mathbf{A}}\mathbf{y} \leq \tilde{\mathbf{b}}$.
The dual is then:

$$\max_{\boldsymbol{\lambda}} \quad \tilde{\mathbf{b}}^{\intercal}\boldsymbol{\lambda} \tag{20a}$$

$$\text{s.t.} \quad \tilde{\mathbf{A}}^{\intercal}\boldsymbol{\lambda} = \mathbf{0} \tag{20b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{20c}$$

As $\boldsymbol{\lambda}$ is a feasible solution to the dual problem, the dual problem is unbounded, and there exist multiple feasible solutions. We are interested in a feasible solution with $\boldsymbol{\lambda} \neq \mathbf{0}$ and therefore add the constraint $\tilde{\mathbf{b}}^{\intercal}\boldsymbol{\lambda} = \mathbf{1}$. Now that the objective function is hidden in the constraints, we can set a different objective function.

The linear program to find minimal infeasible subsystems is then:

$$\max_{\boldsymbol{\lambda}} \quad \sum_i w_i \lambda_i \tag{21a}$$

$$\text{s.t.} \quad \tilde{\mathbf{A}}^\intercal \boldsymbol{\lambda} = \mathbf{0} \tag{21b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{21c}$$

$$\tilde{\mathbf{b}}^\intercal \boldsymbol{\lambda} = \mathbf{1} \tag{21d}$$

where w_i is the weight corresponding to the dual variable λ_i. The support of each solution at a vertex of the feasible region of Problem (21) defines a minimal infeasible subsystem. By changing the weights in the objective, several MISs can be obtained for one infeasible MP solution.

1.4.3 Intersection Cuts

Suppose we are given a problem of the form:

$$\min_{\mathbf{x}} \quad \mathbf{c}^\intercal \mathbf{x} \tag{22a}$$

$$\text{s.t.} \quad \mathbf{x} \in S \cap P \tag{22b}$$

where $\mathbf{c} \in \mathbb{R}^n$, P is a polyhedron, and $S \subset \mathbb{R}^n$ is a closed, potentially non-convex set. In the LP relaxation of Problem (22a), the constraint set becomes $\mathbf{x} \in P$. However, the relaxation might not be a good approximation of the true feasible region. The polyhedral approximation can be tightened by adding intersection cuts [5].

Let $\tilde{\mathbf{x}}$ be a relaxed solution that is infeasible in the original problem: $\tilde{\mathbf{x}} \notin S$. The *conic relaxation* at vertex $\tilde{\mathbf{x}}$ is a cone with apex $\tilde{\mathbf{x}}$ and the neighboring edges are the extreme rays. For a given set S, a convex set C is *S-free* if: $S \cap \operatorname{int}(C) = \emptyset$ [5]. The hyperplane intersecting the boundary of C and the conic relaxation is the *intersection cut* and when added to the problem will cut off $\tilde{\mathbf{x}}$. Due to the convexity of the feasible region and given that the cutoff area lies in C, all feasible solutions remain in the resulting polyhedron [24]. An S-free set C is *maximal* if $C \not\subset C'$ for any S-free set C' [5]. In order to generate deep cuts, the S-free set should be maximal.

As an example, we derive an intersection cut for Problem (10). A visualization is shown in Figure 6. The optimal solution of the relaxed linear program to Problem (10) is $\tilde{\mathbf{x}} = (2, 3.5)$. The set S contains all integer points: $S = \mathbb{Z}$. A possible S-free set C is a disk where the center is $\tilde{\mathbf{x}}$ and the radius is the distance between $\tilde{\mathbf{x}}$ and the closest integer, in our case 0.5. The conic relaxation is the cone with apex $\tilde{\mathbf{x}}$ and the rays correspond to the active constraints at $\tilde{\mathbf{x}}$: $y \leq 1.5x + 0.5$ and

$y \leq -0.5x + 4.5$. The intersection between the circle C and the conic relaxation are the points $P_1 \approx (1.7, 3.1)$ and $P_2 \approx (2.5, 3.3)$. The area between $\tilde{\boldsymbol{x}}$ and the line going through P_1 and P_2 is cut off by adding a constraint to the relaxed LP of Problem (10).

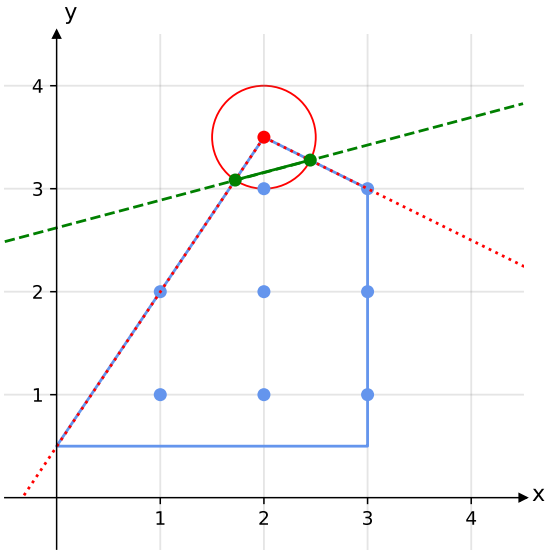


Figure 6: Diagram of intersection cut for Problem (10)

The feasible points of the MIP are the integer points respecting the polyhedral constraints (blue points). The set of feasible solutions to the relaxed LP are all the points in the interior or on the boundary of the polytope (blue line segments). The optimal solution $\tilde{\mathbf{x}}$ to the relaxed LP is the point in the feasible region with the biggest y-value (red point). A possible S-free set C is a disk with center $\tilde{\mathbf{x}}$ and the radius being the distance between $\tilde{\mathbf{x}}$ and the closest integer (red circle). The conic relaxation are the extreme rays generated by $\tilde{\mathbf{x}}$ which correspond to the two active constraints at $\tilde{\mathbf{x}}$ (is indicated by the dotted, red line). The intersection of the conic relaxation and C are the two points in green. The intersection cut is constructed with the line going through the intersecting points (dashed line in green). The area above the green line will be cut off and $\tilde{\mathbf{x}}$ is no longer a feasible solution.

The conic relaxation at $\tilde{\mathbf{x}}$ is a pointed cone with extreme point $\tilde{\mathbf{x}}$ and can be written as:

$$P' = \{\tilde{\mathbf{x}} + \sum_{j=1}^{n} \lambda_j \mathbf{r}^j : \boldsymbol{\lambda} \geq 0\} \tag{23}$$

or as:

$$P' = \{\mathbf{x} | \tilde{\mathbf{A}}\mathbf{x} \leq \tilde{\mathbf{b}}\} \tag{24}$$

where $\mathbf{r}^j$ are the extreme rays, $\tilde{\mathbf{A}}$ is an invertible $n \times n$ submatrix of $\mathbf{A}$ such that the rows are linearly independent and are a basis for $\tilde{\mathbf{x}}$ [5]. A basis for $\tilde{\mathbf{x}}$ are the nonbasic constraints at $\tilde{\mathbf{x}}$. If a constraint is nonbasic, the corresponding slack variable of the standard form is nonbasic and therefore 0. This implies that the constraint is active.

Going back to Problem (10), the polyhedral presentation of the conic relaxation at $(2, 3.5)$ is:

$$P' = \left\{ (x,y) \middle| \begin{bmatrix} -1.5 & 1 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 0.5 \\ 4.5 \end{bmatrix} \right\}$$

The inverse of $\tilde{\mathbf{A}}$ is

$$\tilde{\mathbf{A}}^{-1} = \begin{bmatrix} -0.5 & 0.5 \\ 0.25 & 0.75 \end{bmatrix}$$

and we get the extreme rays $r_1 = (0.5 - 2.5)$ and $r_2 = (-0.5, -0.75)$.

After deriving the conic relaxation, we have $\tilde{\mathbf{x}} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$ and $\mathbf{r}^j = -\tilde{\mathbf{A}}^{-1}_{*,j}$ [5]. For each extreme ray $\mathbf{r}^j$ there either exists an intersection with the boundary of C in which case $\lambda^*_j > 0$ is the step length or the extreme ray is contained in C and $\lambda^*_j = \infty$. The *intersection cut* is defined as [5]:

$$\sum_{i=1}^{n} (\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \tilde{b}_i)/\lambda^*_i \leq -1 \tag{25}$$

where $\tilde{\mathbf{a}}_{i,*}$ denotes the i-th row of basic constraints $\tilde{\mathbf{A}}$.
If we reformulate Equation (25), we see that it matches the definition of a cut in Equation (14) [5]:

$$\sum_{i=1}^{n} (\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \tilde{b}_i)/\lambda^*_i \leq -1 \tag{26}$$

$$\sum_{i=1}^{n} ((1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} - (1/\lambda^*_i)\tilde{b}_i) \leq -1 \tag{27}$$

$$\sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} - \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \leq -1 \tag{28}$$

$$\sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,*}\mathbf{x} \leq -1 + \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \tag{29}$$

$$\alpha_0 = -1 + \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{b}_i \qquad \alpha_j = \sum_{i=1}^{n} (1/\lambda^*_i)\tilde{\mathbf{a}}_{i,j}\mathbf{x}$$

Chapter 2

Metabolic Networks

Computational systems biology studies biological systems and biological networks by building and analyzing mathematical models that approximate their behavior [33]. In this thesis, we focus on models at the cellular level. Instead of looking at the function of all molecules in a cell individually, as in molecular biology, systems biology studies the entire system and especially the links between the different cellular components [29]. Looking at the system as a whole leads to a different understanding of the cell, as the behavior of the cell depends on the interaction of the different components and cannot be explained by only studying the function of components individually [33]. In cells, the chemical reactions that take place in an organism, known as *metabolism*, determine the central function of the cell [33]. Understanding the metabolism is, therefore, necessary in order to understand cellular behavior. *Metabolites* are small molecules that are involved in metabolic reactions [33]. We differentiate between *internal reactions*, involving only internal metabolites, and *exchange reactions*. Exchange reactions are pseudo-reactions representing the transport of metabolites, e.g. nutrients taken up, or products secreted by the cell [34] . In a reaction, a set of metabolites called *reactants* are converted into a different set of metabolites called *products*. The *stoichiometry* is the relative number of metabolites in a reaction. *Enzymes* are important for cell metabolism as they *catalyze* reactions, which means that they accelerate reactions without being consumed by it.
Directed hypergraphs can be used to model and understand cellular behavior [33]. In a *metabolic network*, the nodes are metabolites and the edges represent reactions. A *genome-scale metabolic model* (GEM) or *enzyme constrained metabolic model* is a graph that contains all known metabolic information of a biological system such as reactions, metabolites and enzymes [30].

The information required to build a GEM comes from a *genome-scale reconstruction*, which is the process of identifying the genome of the organism including different components such as the stoichiometry, reaction direction and the corresponding catalyzing enzymes [34] . Depending on the data and assumptions integrated into a model, a different accuracy is achieved, and different questions can be answered [34].

In this thesis, we focus on constraint-based approaches, also known as *constraint-based reconstruction and analysis* (COBRA). A constraint excludes biologically unrealistic behavior, such as violating the laws of thermodynamics or the conservation of mass [33]. With constraint-based methods, one can predict the rate of each reaction in the metabolic network [33]. The mathematical representation of a metabolic network is described in Section 2.1. The COBRA models relevant to this thesis are presented in Section 2.2.

2.1 Mathematical Representation

A *hypergraph* $\mathscr{H}$ is defined by its set of vertices $\mathscr{V}$ and set of hyperedges $\mathscr{E}$. A hyperedge is a pair $E_i = (H_i, T_i)$ of disjoint subsets of $\mathscr{V}$, where H_i denotes the vertices in the *head* and T_i the vertices in the *tail*. A path is a sequence of vertices and hyperedges $P_{v_i,v_{q+1}} = (v_i, E_i, ..., E_{j-1}, v_j, E_j, ..., E_q, v_{q+1})$ where $v_i \in T_i$, $v_{q+1} \in H_q$ and $v_j \in T_j \cap H_{j-1}$ for all $j = 2, ..., q$. A path is a cycle if v_{q+1} is in the tail of E_i [15]. A metabolic network $\mathcal{N}$ can be modeled as a hypergraph and is represented by the tuple $(\mathcal{M}, \mathcal{R}, \mathbf{S}, \mathbf{l}, \mathbf{u})$, where $\mathcal{M}$ is the set of internal metabolites and $\mathcal{R}$ is the set of reactions. The *stoichiometric matrix* $\mathbf{S} \in \mathbb{R}^{m \times n}$ captures the entire network, where m is the number of internal metabolites and n is the number of reactions, including internal reactions $\mathcal{I}$ and exchange reactions $\mathcal{E}$. Usually, the number of metabolites is much smaller than the number of reactions [33]. The columns of $\mathbf{S}$ correspond to the reactions in the network and the entries are the stoichiometric coefficients for each reaction and capture the mass balance for each metabolite. A negative coefficient indicates that a metabolite is consumed and is in the tail of the hyperedge. Vice versa, if the coefficient is positive, the metabolite is produced and belongs to the head of the hyperedge.
The matrix $\mathbf{S}_\mathcal{I}$ is the submatrix of $\mathbf{S}$ that contains the columns of internal reactions only. The stoichiometric matrix $\mathbf{S}$ relates the flux[1] vector $\mathbf{v}$ to the change in metabolite concentration $\mathbf{x}$: $\frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v}$ [25]. The vector of flux distributions indicates the units of flow through all reactions. The vectors $\mathbf{l}$ and $\mathbf{u}$ capture the lower and upper bounds of the flux for all reactions. If the direction of a reaction is forced in

[1] *Flux*, *reaction rate* and *flux distribution* are used interchangeably throughout the thesis.

one direction by the bounds, the reaction is said to be *irreversible*. Otherwise, it is *reversible*. The metabolic fluxes are given in units of $\frac{\text{mmol}}{\text{gDW}\times\text{h}}$, which denotes the flux of a metabolite normalized to the biomass of the cell.

Figure 7 shows a simplified metabolic network with three exchange reactions $\{r_1, r_4, r_8\}$ and five internal reactions $\{r_2, r_3, r_5, r_6, r_7\}$. The set of internal metabolites is $\{A, B, C, D, E, F\}$. The set of reversible reactions is $\{r_3\}$ and the set of irreversible reactions $\{r_1, r_2, r_4, r_5, r_6, r_7, r_8\}$. All irreversible reactions have to be greater or equal to zero.

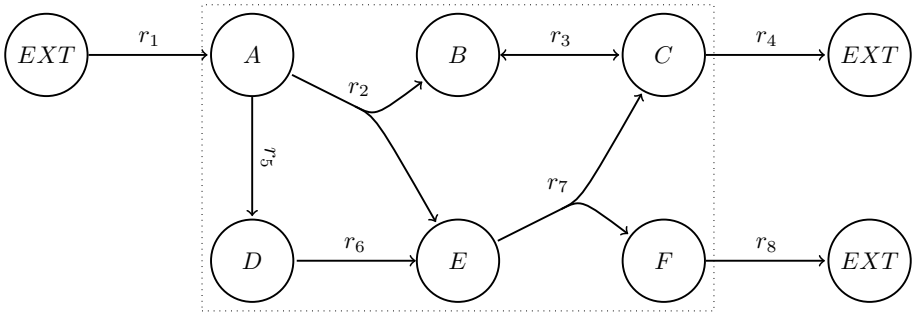


Figure 7: Simplified metabolic network

The corresponding stoichiometric matrix **S** is:

$$\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

S is a 6×8 matrix, where the rows correspond to the internal metabolites in alphabetical order, and the columns denote the reactions r_i in increasing order. In the exchange reaction r_1, A is produced, and therefore the corresponding column has a positive entry for A. In the internal reaction v_2, the consumption of A is captured by a -1 and the production of B and E are denoted by a 1 for each metabolite. In this example, the quantities for the metabolites are either 0, 1 or -1 in all reactions. In more realistic metabolic models, the stoichiometric coefficients are still integer but typically vary.

2.2 Optimization for Metabolic Networks

In dynamic modeling, the dynamics of metabolite concentration are expressed by kinetic laws and kinetic parameters [33]:

$$\frac{dx_i}{dt} = f_i(x_1, x_2, ..., x_n; \Theta) \tag{30}$$

where Θ is the set of kinetic parameters and f_i models how the concentration of x_i changes depending on the concentration of the other metabolites. Such a dynamic model requires a lot of computational data in order to predict the metabolite concentration $\mathbf{x}$.

In constraint-based reconstruction and analysis (COBRA) methods, dynamic metabolic networks are modeled at *steady-state*:

$$\frac{d\mathbf{x}}{dt} = \mathbf{Sv} = \mathbf{0}$$

where $\mathbf{v} \in \mathbb{R}^n$ is the vector of flux distributions and the stoichiometric matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ captures the topology of the metabolic network. As metabolic reactions take place fast compared to other reactions, such as cell division, the steady-state assumption is biologically plausible [38].
In COBRA methods, the set of possible flux distributions is analyzed mathematically, whereas, in dynamic modeling, the metabolite concentration is predicted. With COBRA methods one can predict the flux distribution under different environmental conditions and under changes in the environment [33].
Apart from the steady-state assumption, COBRA methods use context-specific constraints such as *reaction flux bounds* and *mass conservation*. Mass balance is captured in the stoichiometric matrix $\mathbf{S}$. Compared to the dynamic modeling in Equation (30), COBRA methods require much less experimental data [33].

Extreme pathways are a set of vectors that satisfy the model constraints, such that any feasible flux can be written as a convex combination of extreme pathways [32]. There are three types of extreme pathways: *primary systemic pathways* (Type I), *futile cycles* (Type II) and *internal cycles* (Type III). Figure 8 shows the difference between the extreme pathways. Primary systemic pathways are pathways where exchange reactions are used[25]. These pathways are thermodynamically feasible and are desirable steady-state flux solutions (see Section 2.2.4). For most COBRA methods, the goal is to compute a Type I pathway and forbidding Type II and Type III pathways.

In order to understand the difference between Type II and Type III pathways, we have to introduce *currency exchange reactions*. Currency exchange reactions are a subset of internal reactions, where the currency within the cell is changed [25]. For example, a reaction, where ATP is generated from ADP, transfers energy and is a Type II pathway. ATP is a molecule that carries energy. When ATP is converted into ADP, which is of lower energy form, energy is released that can be used in metabolism.
Internal cycles are pathways where none of the exchange reactions and none of the currency exchange reactions are used and are thermodynamically infeasible. Futile cycles are pathways where none of the exchange reactions are used, but at least one currency exchange reaction is used [25]. *Energy-reducing cycles* are futile cycles that are biologically feasible and should be allowed solutions, whereas *energy-generating cycles* (EGCs) are futile cycles that are not wanted in a solution [25]. EGCs are problematic as they can lead to solutions that have a better objective value than when restricting the use of EGCs. One possibility to prevent EGCs is to force the directionality of the reaction by updating its bounds.

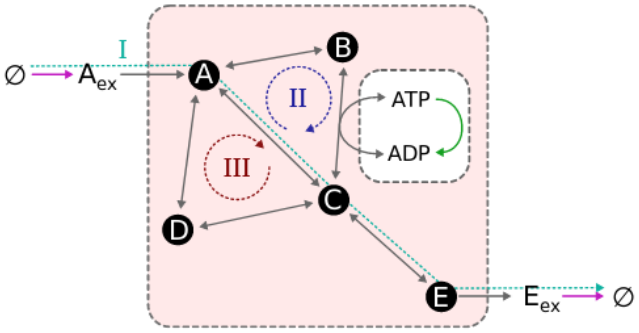


Figure 8: Types of extreme pathways

The figure is taken from [25]. Primary systemic pathways (Type I) are biologically realistic, whereas internal cycles (Type III) are not thermodynamically feasible. Futile cycles (Type II) can either generate energy or reduce energy. Energy-generating cycles are thermodynamically infeasible, whereas energy-reducing cycles are realistic.

The feasible region of a COBRA model usually contains multiple solutions. We are interested in flux distributions that optimize a biological objective. One argument for using mathematical optimization to model the behavior in cells is that a biological objective could be aligned with traits evolutionary selected for [33]. A commonly used objective function is the maximization of growth. Growth can be incorporated into the stoichiometric matrix $\mathbf{S}$ by adding an extra column where the coefficients indicate the contribution to growth [28]. For an overview of biological objectives, see [29].

Depending on the constraints and the objective, we obtain a different COBRA variant. Each corresponds to an optimization problem which can be solved with solution approaches described in Chapter 1. Several COBRA methods relevant to this thesis are explained below.

2.2.1 Flux Balance Analysis

The most basic COBRA method is *flux balance analysis* (FBA). An optimal solution $\mathbf{v}^*$ to an FBA problem, is a flux distribution maximizing a biological linear objective that respects the steady-state assumption and bound constraints.

FBA

$$\max_{\mathbf{v}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{31a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{31b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{31c}$$

where $\mathbf{c} \in \mathbb{R}^n$ determines the linear objective function. The structure of the network is captured in the stoichiometric matrix $\mathbf{S}$. The fluxes $\mathbf{v} \in \mathbb{R}^n$ are bounded by $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$. Constraint (31b) ensures the steady state of the metabolite concentration. We are dealing with a linear program that can be solved efficiently (see Section 1.1).

As an example, let us consider the metabolic network in Figure 9. We have three internal metabolites A, B, C, two irreversible exchange reactions r_1, r_5 and three reversible internal reactions r_2, r_3, r_4. The stoichiometric matrix is:

$$\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix} \tag{32}$$

and we assume the following bounds on the fluxes:

$$\mathbf{l} = (0, -30, -30, -30, 0) \quad \mathbf{u} = (10, 30, 30, 30, 10) \tag{33}$$

If we maximize the flux through the internal reactions, that is $\mathbf{c} = (0, 1, 1, 1, 0)$, an optimal solution is $\mathbf{v}^* = (10, 30, 30, -20, 10)$. The solution contains an internal loop, as there is a flux of 20 going through each of the internal reactions, as seen in Figure 10.

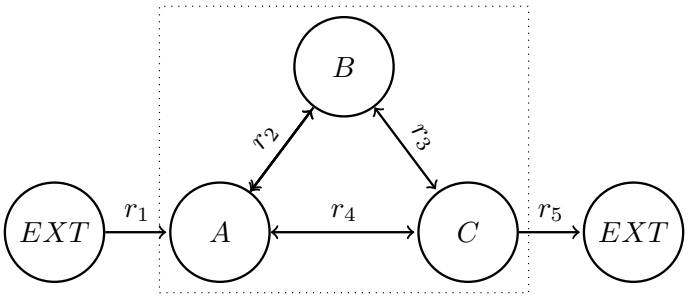


Figure 9: Simple model with internal loop

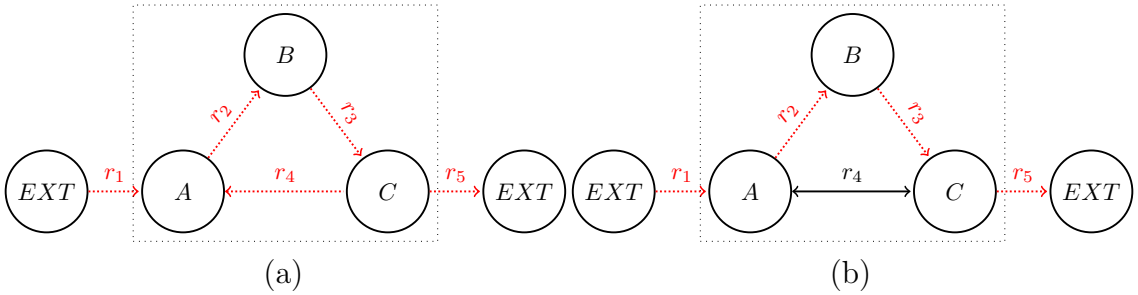


Figure 10: Used reactions in the FBA solution (a) which contains an internal cycle, and used reactions in the ll-FBA solution (b).

A solution to FBA can be a convex combination of primary systematic pathways, futile cycles and internal cycles. To prevent biologically implausible futile cycles, the directionality is restricted with the bound constraints (Constraint (31c)). Internal cycles are biologically not realistic, but are part of the solution space, posing a major drawback to FBA solutions.

2.2.2 CycleFreeFlux

Suppose we are given a solution $\mathbf{v}^{FBA}$ to an FBA problem (Problem (31)). The following linear program, known as *CycleFreeFlux* (CFF), returns a solution $\mathbf{v}$ that is structurally consistent with $\mathbf{v}^{FBA}$ and with the minimum sum of reaction rates such that the objective values are identical [12].

CFF

$$\min_{\mathbf{v}} \quad \sum_i |v_i| = \sum_{i:v_i^{FBA}>0} v_i - \sum_{i:v_i^{FBA}<0} v_i \tag{34a}$$

$$\text{s.t.} \quad \mathbf{Sv} = \mathbf{0} \tag{34b}$$

$$\mathbf{c}^\intercal \mathbf{v} = \mathbf{c}^\intercal \mathbf{v}^{FBA} \tag{34c}$$

$$0 \leq v_i \leq v_i^{FBA} \quad \text{for } i \text{ with } v_i^{FBA} \geq 0 \tag{34d}$$

$$v_i^{FBA} \leq v_i \leq 0 \quad \text{for } i \text{ with } v_i^{FBA} < 0 \tag{34e}$$

$$v_i = v_i^{FBA} \quad \forall i \in \mathcal{E} \tag{34f}$$

Besides the steady-state assumption in Constraint (34b), the reaction direction has to match the direction in $\mathbf{v}^{FBA}$ or become zero, enforced by Constraint (34d) and Constraint (34c). In addition, the reaction rates of the exchange reactions $\mathcal{E}$ remain unchanged (Constraint (34f)). Constraint (34c) ensures that the objective value of $\mathbf{v}$ matches the objective value of $\mathbf{v}^{FBA}$. The objective function can be rewritten by replacing the absolute function with the sum of positive reaction rates minus the sum of negative reaction rates, as the sign of reaction rates is known from $\mathbf{v}^{FBA}$. As Problem (34) is a linear program, it is easy to solve (see Section 1.1.1). If the reactions contained in an internal loop are allowed to be zero, and if the objective does not depend on a reaction that is contained in an internal loop, $\mathbf{v}$ is thermodynamically feasible [25]. We will see in Section 2.2.4 that formulating a mathematical program such that the resulting flux is in the thermodynamically feasible subspace is $\mathcal{NP}$-hard.

Let us go back to the FBA example visualized in Figure 9. If we apply CycleFreeFlux to the FBA solution $\mathbf{v}^{FBA} = (10, 30, 30, -20, 10)$ with $\mathbf{c}^\intercal \mathbf{v}^{FBA} = 40$, the solution flux $\mathbf{v}^* = (10, 30, 30, -20, 10)$ still contains an internal loop.

As explained in [12], the CycleFreeFlux algorithm can be used to enumerate cycles. For each internal reaction i, we solve three LPs. We first solve an FBA problem, where we maximize the flux through reaction i, whose solution is denoted by $\mathbf{v}^{FBA}$. We solve the CFF problem based on $\mathbf{v}^{FBA}$, whose solution is denoted by $\mathbf{v}^{CFF(1)}$. If $\mathbf{v}^{FBA} \neq \mathbf{v}^{CFF(1)}$, we know that $\mathbf{v}^{FBA}$ contains an internal cycle. We solve another CFF problem based on $\mathbf{v}^{FBA}$ with the additional constraint $v_i = v_i^{FBA}$, whose solution is denoted by $\mathbf{v}^{CFF(2)}$. The solution $\mathbf{v}^{CFF(1)}$ contains one internal cycle, which corresponds to the difference between $\mathbf{v}^{CFF(2)} - \mathbf{v}^{CFF(1)}$. Alternatively, we check whether the difference of $\mathbf{v}^{FBA} - \mathbf{v}^{CFF(1)}$ contains an internal loop directly.

2.2.3 Thermodynamic Flux Balance Analysis

The second law of thermodynamics states that the entropy in a closed system cannot decrease. In a metabolic network, each reaction i is associated with a scalar $\Delta\mu_i$ denoting the *Gibbs free energy change* or *potential difference* of the reaction. The following holds [37]:

$$\Delta\mu_i = \Delta\mu_i^\circ + RT\ln(Q_i)$$

R is the universal gas constant, which relates energy to the amount of substance and temperature, and T is the temperature in Kelvin. Q_i is the reaction quotient and denotes the ratio of product concentration to reactant concentration. The variable $\Delta\mu_i^\circ$ is the *standard chemical potential*, or *equilibrium potential*, or *standard Gibbs free energy* of reaction i. Instead of using the reaction quotient, we can calculate the concentration of reactants and products in reaction i by using the stoichiometric coefficients $s_{i,*}$ and the metabolite concentration $\mathbf{c}$ [25]:

$$\Delta\mu_i = \Delta\mu_i^\circ + RT\sum_i s_{i,*}^\intercal \ln(\mathbf{c})$$

where $\Delta\mu_i$ is the change of Gibbs free energy for reaction i.
Written in matrix notation, we have:

$$\boldsymbol{\Delta\mu} = \boldsymbol{\Delta\mu}^\circ + RT\mathbf{S}^\intercal\ln(\mathbf{c}) \tag{35}$$

One method that integrates thermodynamic data to get thermodynamically feasible solutions is *thermodynamic flux balance analysis* (TFBA).

TFBA

$$\begin{aligned}
\max_{\mathbf{v},\boldsymbol{a},\mathbf{x},\boldsymbol{\mu}} \quad & \mathbf{c}^\intercal\mathbf{v} && \text{(36a)}\\
\text{s.t.} \quad & \mathbf{S}\mathbf{v} = \mathbf{0} && \text{(36b)}\\
& \mathbf{l} \le \mathbf{v} \le \mathbf{u} && \text{(36c)}\\
& 0 \le Ma_i - \mathbf{v}_i \le M \quad \forall i \in \mathcal{I} && \text{(36d)}\\
& \epsilon \le Ma_i + \boldsymbol{\Delta\mu_i} \le M - \epsilon \quad \forall i \in \mathcal{I} && \text{(36e)}\\
& \boldsymbol{\Delta\mu} = \boldsymbol{\Delta\mu}^\circ + RT\mathbf{S}_\mathcal{I}^\intercal\mathbf{x} && \text{(36f)}\\
& \ln(\mathbf{b}^L) \le \mathbf{c} \le \ln(\mathbf{b}^U) && \text{(36g)}\\
& a_i \in \{0,1\} \quad \forall i \in \mathcal{I} && \text{(36h)}
\end{aligned}$$

The steady-state constraint (Constraint (36b)), the bound constraints (Constraint (36c)) and the linear objective function corresponding to the FBA model (Problem (31)). Constraint (36d) and Constraint (36e) ensure that the Gibbs free energy is never reduced for any reaction i: $v_i = 0 \vee \text{sign}(v_i) = -\text{sign}(\Delta\mu_i)$. $M \in \mathbb{R}$ is a large constant that does not restrict the value of $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$. The binary variables $\boldsymbol{a}$ indicate the direction of $\mathbf{v}$: $a_i = 1$ implies $v_i > 0$ and $a_i = 0$ implies $v_i < 0$. If $a_i = 1$, v_i is zero or positive as $-M \leq -v_i \leq M$ holds. The variable $\Delta\mu_i$ is not allowed to be zero so it is bounded by a small value ϵ. In that case, $\Delta\mu_i$ is negative as $\epsilon - M \leq \Delta\mu_i \leq -\epsilon$. If $a_i = 0$, $0 \leq -v_i \leq M$ holds and the flux v_i is zero or negative. The variable $\Delta\mu_i$ is positive as $\epsilon \leq \Delta\mu_i \leq M - \epsilon$. $\mathbf{c}$ is the vector of metabolite concentrations which is bounded by $\ln(\mathbf{b}^L)$ and $\ln(\mathbf{b}^U)$ [25].

A solution to TFBA is thermodynamically feasible, however, the model requires the standard equilibrium potential for each reaction $\boldsymbol{\Delta\mu}^\circ$ which is usually not known. Additionally, the resulting problem is a mixed-integer problem and more difficult to solve than the FBA problem.

2.2.4 Loopless FBA

A simpler method to ensure that a solution does not contain an internal loop, which does not require additional thermodynamic data is *loopless FBA* (ll-FBA). To respect the second law of thermodynamics, the following inequality for reaction i is required:

$$\{v_i = 0\} \vee \{\text{sign}(v_i) = -\text{sign}(\Delta\mu_i)\} \tag{37}$$

which means that if a reaction carries flux, Gibbs free energy decreases [23]. A *loop* is a nonzero flux vector $\boldsymbol{\ell}$ such that the internal network is at steady-state: $\mathbf{S}_\mathcal{I}\boldsymbol{\ell} = \mathbf{0}$ [26]. As ℓ_i and $\Delta\mu_i$ have to be of opposite sign, unless $\ell_i = 0$, we know that $\boldsymbol{\ell}^\intercal\boldsymbol{\Delta\mu} \neq 0$. However, *flux balance*, or *Kirchhoff's second law*, states that the reaction energies around a circuit have to sum up to zero, and therefore $\boldsymbol{\ell}$ is not thermodynamically feasible [37]. A flux distribution $\mathbf{v}$ contains a loop if there exists a nonzero vector $\boldsymbol{\ell}$ with $\mathbf{S}_\mathcal{I}\boldsymbol{\ell} = \mathbf{0}$ such that:

$$\text{sign}(\ell_i) \in \{\text{sign}(v_i), 0\} \quad \forall i \in \mathcal{I}$$

For a solution to be thermodynamically feasible, the reaction energies around any flux $\mathbf{v}$ have to sum up to zero: $\mathbf{v}^\intercal\boldsymbol{\Delta\mu} = 0$, where $\boldsymbol{\Delta\mu}$ is the vector of Gibbs free energy [37]. As we are interested in internal cycles, it is sufficient to verify that $\mathbf{v}_\mathcal{I}^\intercal\boldsymbol{\Delta\mu} = 0$, where $\mathbf{v}_\mathcal{I}$ is a flux distribution through internal reactions and $\boldsymbol{\Delta\mu}$ are the corresponding changes in Gibbs free energy. Any steady-state nonzero pathway that

just uses internal reactions is a loop: $\mathbf{S}_\mathcal{I}\mathbf{v}_\mathcal{I} = \mathbf{0}$ [26]. The nullspace of $\mathbf{S}_\mathcal{I}$ is defined as $\text{null}(\mathbf{S}_\mathcal{I}) := \{\mathbf{x} \in \mathbb{R}^{|\mathcal{I}|} : \mathbf{S}_\mathcal{I}\mathbf{x} = \mathbf{0}\}$. Let $\mathbf{B} \in \mathbb{R}^{|\mathcal{I}|\times k}$ be a matrix with columns formed by the set of vectors $\{\mathbf{b}_i\}_{i=1}^k$ that form a basis for $\text{null}(\mathbf{S}_\mathcal{I})$. Any loop $\boldsymbol{\ell}$ lies in the nullspace of $\mathbf{S}_\mathcal{I}$ and can thus be written as $\boldsymbol{\ell} = \mathbf{B}^\intercal\boldsymbol{\alpha}$, where coefficients $\alpha_i \in \mathbb{R}$ [37]. Therefore, the following removes solutions with internal loops: $\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0}$. Adding the loopless constraints to the FBA problem, we obtain the following mathematical program:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{38a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{38b}$$

$$\mathbf{l} \le \mathbf{v} \le \mathbf{u} \tag{38c}$$

$$\Delta\mu_i v_i < 0 \vee v_i = 0 \quad \forall i \in \mathcal{I} \tag{38d}$$

$$\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0} \tag{38e}$$

where $\mathbf{v} \in \mathbb{R}^n$ and $\boldsymbol{\Delta\mu} \in \mathbb{R}^{|\mathcal{I}|}$. For readability, index i corresponds to the pair of v_i and $\Delta\mu_i$, even though $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$ are of different lengths.
We can write Constraint (38e) as $\boldsymbol{\Delta\mu} = \mathbf{S}^\intercal\boldsymbol{\mu}$ [26, 37, 23]. Let us define the vector of potential differences of the internal reactions as $\boldsymbol{\Delta\mu} = \mathbf{S}_\mathcal{I}^\intercal\boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is approximately the chemical potential. Each vector in the rowspace of $\mathbf{S}_\mathcal{I}$ is orthogonal to each vector in the nullspace of $\text{null}(\mathbf{S}_\mathcal{I})$ [26]. Therefore, $\mathbf{B}^\intercal\boldsymbol{\Delta\mu} = \mathbf{0}$.
Rewriting Constraint (38e), we obtain the following mathematical program [23]:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{39a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{39b}$$

$$\mathbf{l} \le \mathbf{v} \le \mathbf{u} \tag{39c}$$

$$\Delta\mu_i v_i < 0 \vee v_i = 0 \quad \forall i \in \mathcal{I} \tag{39d}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_\mathcal{I}^\intercal\boldsymbol{\mu} \tag{39e}$$

where $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\Delta\mu} \in \mathbb{R}^{|\mathcal{I}|}$. Problem (39) is $\mathcal{NP}$-hard and much more complicated to solve than FBA [35]. We use $\boldsymbol{\Delta\mu}$ and $\boldsymbol{\mu}$ due to the biological interpretability, however, it is not necessary to define $\boldsymbol{\Delta\mu}$ and use the right-hand side of Constraint (39e) in Constraint (39d).

Constraints (38e) and (39e) pose a challenge due to the disjunction, the strict equality and the product of decision variables $\boldsymbol{\Delta\mu}$ and $\mathbf{v}$. To simplify the model, the disjunction is rewritten and we arrive at the loopless FBA model used in this thesis:

ll-FBA

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal\mathbf{v} \tag{40a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{40b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{40c}$$

$$\Big((v_i \geq 0) \wedge (\Delta\mu_i \leq -\epsilon)\Big) \vee \Big((v_i \leq 0) \wedge (\Delta\mu_i \geq \epsilon)\Big) \qquad \forall i \in \mathcal{I} \tag{40d}$$

$$\boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal\mathbf{S}_\mathcal{I} \tag{40e}$$

Whereas the value of $\boldsymbol{\Delta\mu}$ in TFBA corresponds to the Gibbs free energy change tested in experiments, in ll-FBA only the sign($\boldsymbol{\Delta\mu}$) corresponds to the actual Gibbs free energy change [37]. As the value does not matter, we can scale $\boldsymbol{\Delta\mu}$ by not allowing it to be in the interval $[-\epsilon, \epsilon]$. The resulting problem is a disjunctive program with linear constraints in the disjunctions and can be reformulated as a mixed-integer program (see Section 1.3). The focus of this thesis is to analyse the running time of ll-FBA and compare different solution approaches to solve ll-FBA. A cutting plane approach to solve ll-FBA was also studied by [35].

Going back to the previous example with $\mathbf{S}$ defined in Equation (32) and flux bounds defined in Equation (33), the loopless flux distribution is $\mathbf{v}^* = (10, 10, 10, 0, 10)$ with $\mathbf{c}^\intercal\mathbf{v}^* = 20$ which is shown in Figure 10.

In ll-FBA internal cycles are excluded from the solution space. However, energy-generating cycles still have to be blocked by restricting the directionality.

2.2.5 st-FBA

Semi-thermodynamic Flux Balance Analysis (st-FBA) combines the ll-FBA and TFBA approaches [25]. The loopless FBA formulation is extended by constraining the Gibbs free energy change $\boldsymbol{\Delta\mu}$ of a subset of reactions. Metabolites that carry energy such as ATP and their related compounds such as ADP and AMP are treated as in TFBA. The chemical potential of such a metabolite is limited by bounds known from experiments. Bounding the chemical potential of these energy-carrying currency metabolites excludes energy-generating cycles, and it is no longer required to force the direction of certain reactions as in ll-FBA. st-FBA is thus a trade-off between the $\boldsymbol{\Delta\mu}$

values corresponding to the Gibbs free energy changes which requires experimental data, and the approximation of the Gibbs Free energy changes which only deals with the directionality. We require more experimental data than for ll-FBA, as we need the bounds of reactions involved in energy-reducing cycles. As we add the experimental data just on a small subset of reactions, just a fraction of information needed for TFBA is needed for st-FBA.

2.2.6 Enzyme Constrained Metabolic Models

In FBA, the optimal reaction rate of a metabolic network is limited by the nutrition uptake. However, the flux depends also on the enzyme abundances catalyzing the reactions. Especially modeling physiological responses, such as *overflow metabolism*, requires enzymatic data [36]. Overflow metabolism refers to the phenomenon that a cell uses fermentation instead of respiration, even though respiration is more energetically efficient [3]. There are several hypotheses to explain the metabolic switch from respiration to fermentation. One hypothesis is that overflow metabolism is connected to enzyme capacity limitations. When using fermentation, the cell does not grow as fast as with respiration, but it yields more energy with the same enzyme mass [36]. *GECKO* is one method that integrates enzyme constraints into a genome-scale model. The following section is based on [36].

Suppose reaction j is catalyzed by enzyme E_i. The *turnover number* indicates how fast an enzyme is. The reaction rate v_j depends on the *enzyme usage* e_i and the turnover number k_{cat}^{ij}:

$$k_{cat}^{ij} e_i = v_j \tag{41}$$

The turnover number is given in $\frac{1}{\mathrm{h}}$, the enzyme usage in $\frac{\mathrm{mmol}}{\mathrm{gDW}}$, such that $\mathbf{v}$ has the unit of $\frac{\mathrm{mmol}}{\mathrm{gDW} \times \mathrm{h}}$. Equation (41) is also known as *enzyme mass balance*. To account for enzyme mass balance in a GEM, matrix S^{GECKO} is constructed by extending the stoichiometric matrix $\mathbf{S}$ by three submatrices. A row is added for each enzyme E_i and a column for the corresponding enzyme usage e_i, with p enzymes. The lower left submatrix contains the enzyme information on the diagonal, that is $-1/k_{cat}^{ij}$. The lower right matrix is the identity matrix.

The upper right matrix contains only zeros. The resulting matrix is of the form:

$$S^{GECKO} = \left[\begin{array}{ccc|ccc} s_{1,1} & \dots & s_{1,n} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & \dots & s_{m,n} & 0 & \dots & 0 \\ \hline -1/k_{cat}^{11} & \dots & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -1/k_{cat}^{pn} & 0 & \dots & 1 \end{array}\right] = \left[\begin{array}{cc} \mathbf{S} & \mathbf{0}_{m,p} \\ \text{diag}(-1/k_{cat}^{ij}) & \mathbf{I}_p \end{array}\right]$$

In addition to the reaction rates $\mathbf{v}$, we are interested in the enzyme usage $\mathbf{e}$. We obtain the following optimization problem:

$$\max_{\mathbf{v},\mathbf{e}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{42a}$$

$$\text{s.t.} \quad S^{GECKO}(\mathbf{v}, \mathbf{e}) = \mathbf{0} \tag{42b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{42c}$$

$$\mathbf{0} \leq \mathbf{e} \leq [\mathbf{E}] \tag{42d}$$

where $[E_i]$ is the intracellular concentration of enzyme E_i. The objective function and Constraint (42c) are also part of the FBA problem. Constraint (42b) forces a steady state of the metabolites and enzyme mass balance. As the upper right matrix contains only zeros, the steady-state constraint $\mathbf{Sv} = \mathbf{0}$ is preserved. In addition to bounds on the fluxes, Constraint (42d) limits the enzyme usage e_i, which cannot be negative and has to be below the intracellular enzyme concentration $[E_i]$:

$$0 \leq e_i \leq [E_i]$$

Together both lower submatrices of S^{GECKO} make up the enzyme mass balance constraints:

$$\begin{aligned} -\frac{1}{k_{cat}^{ij}} v_j + e_i &= 0 \\ e_i &= \frac{1}{k_{cat}^{ij}} v_j \\ k_{cat}^{ij} e_i &= v_j \end{aligned}$$

Depending on the type of enzyme, S^{GECKO} is modified slightly. If different enzymes catalyze the same reaction, they are called *isozymes* and one mass balance constraint for each isozyme is added to the model.

Usually, the total enzyme abundance is limited by including the following constraint:

$$\sum_{i}^{p} \mathrm{MW}_i e_i \leq P$$

where MW_i is the molecular weight of enzyme i and P is the total protein content in the cell [36].

Chapter 3

Methods and Instances

In Chapter 2, we have seen how mathematical optimization can be used to predict fluxes $\mathbf{v}$ in a cell that optimize a biological objective. In particular ll-FBA (Problem (40)) can be used to predict fluxes that do not contain internal loops. However, ll-FBA is difficult to solve due to the constraint that $\Delta\mu_i$ and v_i have to be of opposite sign for each internal reaction i, unless $v_i = 0$, which is modeled by a disjunction. First, we explain briefly how we test for a solution whether it does not contain internal loops (Section 3.1). Then, we introduce different reformulations of the ll-FBA problem including indicator constraints and big-M constraints (Section 3.2). We want to compare solving the reformulations of ll-FBA directly to

1. blocking cycles in the big-M reformulation,
2. decomposing the problem,
3. solving a relaxed problem with cuts and
4. solving the convex-hull formulation.

For 1., we take a solution to the FBA problem (Problem (31)) which may contain internal loops. Loops are identified and cuts are added to the ll-FBA formulation to exclude solutions that contain a specific loop from the feasible region (Section 3.3). For 2., we decompose the problem and solve it with no-good cuts and combinatorial Benders' cuts. We solve a relaxed version of ll-FBA, where we solve the FBA problem, but assign binary variables dependent on the directionality of $\mathbf{v}$. If a relaxed solution is not a feasible solution to the ll-FBA problem, a cut is added to the relaxed problem that cuts off the assignment of binary variables. This procedure is repeated until a relaxed solution is a feasible solution to ll-FBA (Section 3.4).

Next, we generate stronger cuts than a no-good cut using combinatorial Benders' cuts. We exploit the duality of the infeasible subproblem to generate cuts that include only a subset of binary variables (Section 3.5).
For 3., we experiment with intersection cuts (Section 3.6), where the feasible region is split into the polyhedral set P, which is defined by the inequality and equality constraints, and the set S, which is defined by the disjunctions. An S-free set C is defined around a relaxed solution $\tilde{\mathbf{x}}$ that lies in P but not in S. The intersection cut is the hyperplane going through the intersection of the conic relaxation at $\tilde{\mathbf{x}}$ with C.
For 4., the ll-FBA problem is written as a disjunctive program, which includes a binary variable for each disjunct. The problem is then solved by using the big-M reformulation and the convex-hull formulation (Section 3.7).
In Section 3.8 the data we use for the experiments is presented.

3.1 Feasibility Test

For a given flux distribution $\tilde{\mathbf{v}}$ and the corresponding directions $\tilde{\boldsymbol{a}}$ the following problem is feasible if it is loopless:

$$\max_{\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad 0 \tag{43a}$$

$$\text{s.t.} \quad -M \leq \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} : \tilde{a}_i = 1 \tag{43b}$$

$$\epsilon \leq \Delta\mu_i \leq M \quad \forall i \in \mathcal{I} : \tilde{a}_i = 0 \tag{43c}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^{\intercal}\boldsymbol{\mu} \tag{43d}$$

where we set the big-M constant to 1000 and ϵ to 1.

Note that if $\tilde{v}_i = 0$, the reaction can be removed from the problem by removing the reaction from $\mathbf{S}_{\mathcal{I}}$ and without adding variable $\Delta\mu_i$. Problem (43) can also be used to test the looplessness of a solution on a subset of internal reactions by setting the flux of all other reactions to zero.

Going back to the example visualized in Figure 9 in Section 2.2.1, for the solution $\tilde{\mathbf{v}} = (10, 30, 30, -20, 10)$ where $\tilde{v}_2, \tilde{v}_3, \tilde{v}_4$ are the fluxes through the internal reactions, and $\tilde{\boldsymbol{a}} = (1, 1, 0)$. The solution contains a loop and Problem (43) is infeasible.
For the loopless solution $\tilde{\mathbf{v}} = (10, 10, 10, 0, 10)$, we only look at the nonzero internal fluxes $\tilde{v}_2, \tilde{v}_3$ and the corresponding directionality is $\tilde{\boldsymbol{a}} = (1, 1)$. Problem (43) is feasible, and the solution is $\boldsymbol{\Delta\mu} = (-1, -1)$ and $\boldsymbol{\mu} = (2, 1, 0)$.

3.2 Loopless FBA Formulations

We can reformulate ll-FBA (Problem (40)) to no longer write the disjunctions explicitly. The mixed-integer program of flux balance analysis without unbounded internal cycles using indicator constraints is given by [37]:

ll-FBA (indicator)

$$\max_{\mathbf{v},\boldsymbol{a},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{44a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{44b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{44c}$$

$$a_i = 1 \implies v_i \geq 0 \quad \forall i \in \mathcal{I} \tag{44d}$$

$$a_i = 1 \implies \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} \tag{44e}$$

$$a_i = 0 \implies v_i \leq 0 \quad \forall i \in \mathcal{I} \tag{44f}$$

$$a_i = 0 \implies \Delta\mu_i \geq \epsilon \quad \forall i \in \mathcal{I} \tag{44g}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^\intercal \boldsymbol{\mu} \tag{44h}$$

$$a_i \in \{0,1\} \quad \forall i \in \mathcal{I} \tag{44i}$$

The mixed-integer program of flux balance analysis without unbounded internal cycles using big-M constraints takes the form [37]:

ll-FBA (big-M)

$$\max_{\mathbf{v},\boldsymbol{a},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{45a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{45b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{45c}$$

$$-Ma_i + \epsilon(1 - a_i) \leq \Delta\mu_i \leq -\epsilon a_i + M(1 - a_i) \quad \forall i \in \mathcal{I} \tag{45d}$$

$$-M(1 - a_i) \leq v_i \leq Ma_i \quad \forall i \in \mathcal{I} \tag{45e}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^\intercal \boldsymbol{\mu} \tag{45f}$$

$$a_i \in \{0,1\} \quad \forall i \in \mathcal{I} \tag{45g}$$

The big-M formulation ensures the opposite sign of v_i and $\Delta\mu_i$ if $v_i \neq 0$. If the flux through reaction i is a forward flux, that is $v_i > 0$ and $a_i = 1$, it holds that $-M \leq \Delta\mu_i \leq -\epsilon$ and analogously for a backward flux. We set ϵ to 1 to match the formulation of [37]. The value of ϵ affects the scale of $\boldsymbol{\Delta\mu}$ (and $\boldsymbol{\mu}$). The value of ϵ should be bigger than the solver precision to differentiate it from 0.

The big-M constant is the maximal absolute value of the flux bounds $\mathbf{l}, \mathbf{u}$. As explained in Section 2.2.4, Constraints (44h) and (45f) can be replaced by $\mathbf{B}^\intercal \boldsymbol{\Delta\mu} = \mathbf{0}$. The resulting model is then:

ll-FBA (nullspace)

$$\max_{\mathbf{v},\boldsymbol{a},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{46a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{46b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{46c}$$

$$-Ma_i + \epsilon(1 - a_i) \leq \Delta\mu_i \leq -\epsilon a_i + M(1 - a_i) \quad \forall i \in \mathcal{I} \tag{46d}$$

$$-M(1 - a_i) \leq v_i \leq Ma_i \quad \forall i \in \mathcal{I} \tag{46e}$$

$$\mathbf{B}^\intercal \boldsymbol{\Delta\mu} = \mathbf{0} \tag{46f}$$

$$a_i \in \{0, 1\} \quad \forall i \in \mathcal{I} \tag{46g}$$

3.3 Blocking Cycles

We use the CycleFreeFlux algorithm to enumerate cycles as explained in Section 2.2.2. Suppose we have a solution that contains a loop $\boldsymbol{\ell}$ and let $\boldsymbol{a}$ be the corresponding binary variables indicating the direction of the flux. We can write the assignment of binary variables as Boolean expression: $t_1 \wedge ... \wedge t_i \wedge ...t_{|\mathcal{I}|}$, where $t_i = a_i$ if $a_i = 1$ and otherwise $t_i = 1 - a_i$. To block the cycle, the assignment of the a_i's has to be constrained: $\neg(t_1 \wedge ... \wedge t_i \wedge ...t_{|\mathcal{I}|}) = 1$. Formulating the expression as a linear constraint yields: $t_1 + ... + t_i + ...t_{|\mathcal{I}|} \geq 1$.

A detected cycle can be blocked by a cut in the loopless FBA problem. The solution will not be impacted, but the running time can be affected. Instead of blocking all detected cycles, we experiment with blocking a subset of cycles and by blocking the smallest cycles, which potentially leads to stronger cuts.

3.4 Decomposition with No-Good Cuts

When using no-good cuts to solve ll-FBA, the solving procedure is divided into first solving the relaxed problem and then checking the feasibility of a relaxed solution. The following relaxed version of ll-FBA (indicator) (Problem (44)) is solved, where the loopless constraints are ignored, but the indicator variables $\boldsymbol{a}$ are assigned:

$$\max_{\mathbf{v},\boldsymbol{a}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{47a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{47b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{47c}$$

$$a_i = 1 \implies v_i \geq 0 \quad \forall i \in \mathcal{I} \tag{47d}$$

$$a_i = 0 \implies v_i \leq 0 \quad \forall i \in \mathcal{I} \tag{47e}$$

The indicator constraints can be linearized by using big-M constraints. If a relaxed solution $\mathbf{x}^{IP}$ is thermodynamically infeasible, the following constraint is added:

$$\sum_{i \in \mathcal{I}: x_i^{IP}=1} a_i + \sum_{i \in \mathcal{I}: x_i^{IP}=0} (1 - a_i) \leq |\mathcal{I}| - 1 \tag{48}$$

The problem is now solved again and this procedure is repeated until a relaxed solution is thermodynamically feasible.

As an example, let us consider the metabolic network in Figure 11. We have four internal metabolites A, B, C, D, two irreversible exchange reactions r_1, r_5, two irreversible internal reactions r_6, r_7 and three reversible internal reactions r_2, r_3, r_4. We assume the following bounds on the fluxes:

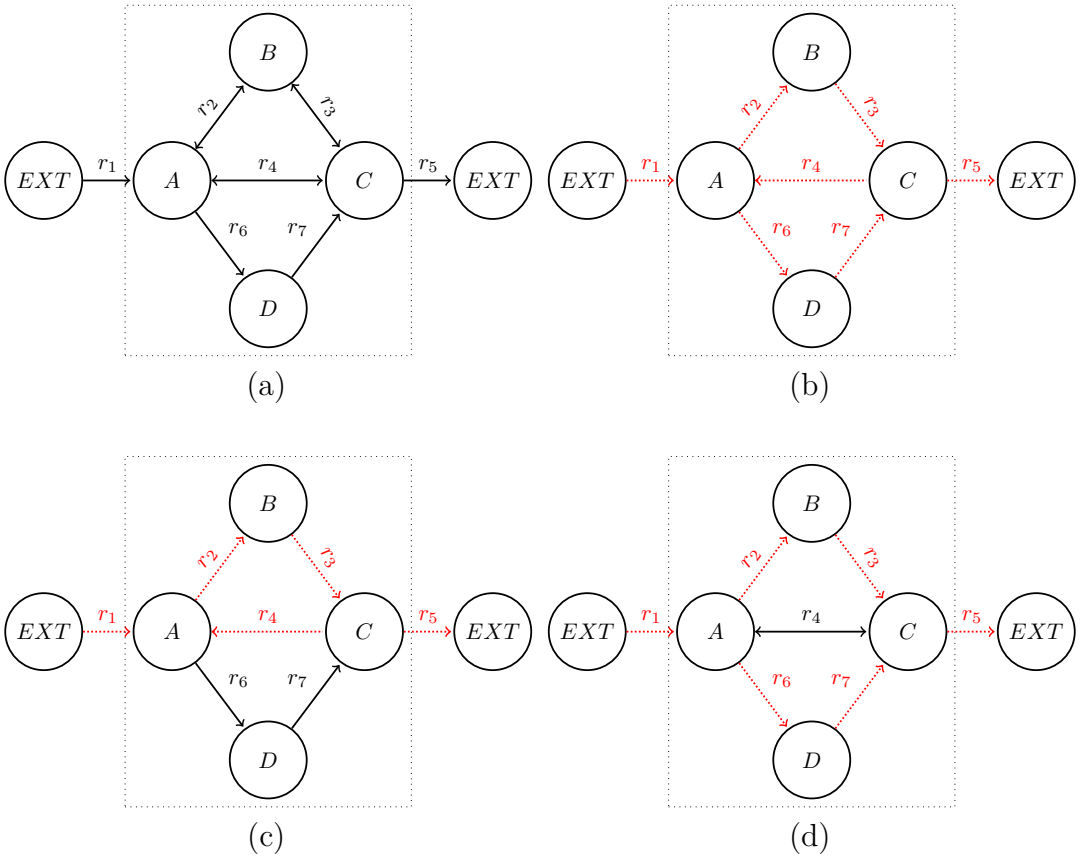
$$\mathbf{l} = (0, -10, -10, -20, 0, 0, 0) \quad \mathbf{u} = (20, 30, 30, 30, 20, 10, 10)$$

If we maximize the flux through all reactions, that is $\mathbf{c} = \mathbf{1}$, the relaxed solution is $\mathbf{v}^* = (20, 30, 30, -20, 20, 10, 10)$ and $\boldsymbol{a}^* = (1, 1, 0, 1, 1)$. The relaxed solution is not thermodynamically feasible as it contains two internal loops. The corresponding no-good cut is:

$$(1 - a_3) + \sum_{i \in \{1,2,4,5\}} a_i \leq 5 - 1$$

After adding the no-good cut, the relaxed solution is $\mathbf{v}^* = (20, 30, 30, -10, 20, 0, 0)$. After 5 iterations, the solution found is feasible for ll-FBA.

Figure 11: Simple model with two internal loops



The reactions used in the FBA solution with two internal loops (b), the reactions used in a relaxed solution with one internal loop (c), the ll-FBA solution without flux through the internal cycles (d).

3.5 Combinatorial Benders' Decomposition

When deriving combinatorial Benders' cuts for ll-FBA, the problem is decomposed into a master problem and a subproblem. The solving procedure is divided into first solving the relaxed problem and then checking the feasibility of the relaxed solution, similar to the no-good cut derivation. However, instead of forbidding the infeasible relaxed solution, we aim to identify the variables that lead to thermodynamic infeasibility and potentially derive stronger cuts.

The ll-FBA problem matches the required problem form (compare to Problem (16)). We have a linear objective $\mathbf{c}^\intercal\mathbf{v}$, continuous variables $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$, and a set of binary variables $\boldsymbol{a}$. There are linear constraints on the continuous variables Constraints (44b), (44c) and (44h), and no constraints on the binary variables. The continuous variables and the binary variables are only connected by the indicator constraints (Constraints (44d–g)), and we can split the MIP into a master and a subproblem.

Problem (47) is the master problem (MP) and the subproblem (SP) is the following parameterised program:

$$\max_{\boldsymbol{\mu},\boldsymbol{\Delta\mu}} \quad 0 \tag{49a}$$

$$\text{s.t.} \quad a_i^{MP} = 1 \implies \Delta\mu_i \leq -\epsilon \quad \forall i \in \mathcal{I} \tag{49b}$$

$$a_i^{MP} = 0 \implies \Delta\mu_i \geq \epsilon \quad \forall i \in \mathcal{I} \tag{49c}$$

$$\boldsymbol{\Delta\mu} = \mathbf{S}_{\mathcal{I}}^{\intercal}\boldsymbol{\mu} \tag{49d}$$

where $\boldsymbol{a}^{MP}$ are the values of a solution to the master problem. If the subproblem is infeasible, we compute a corresponding minimal infeasible subsystem (MIS) $\mathcal{C}$. The minimal infeasible subsystem contains a minimal number of constraint indices that make the subproblem infeasible. The following combinatorial Benders' (CB) cut is added to the master problem if the subproblem is infeasible:

$$\sum_{i \in \mathcal{C}: a_i^{MP}=0} a_i + \sum_{i \in \mathcal{C}: a_i^{MP}=1} (1 - a_i) \geq 1$$

Note that if $\mathcal{C}$ contains all internal reactions $\mathcal{I}$, the CB cut corresponds to a no-good cut (Equation (48)).

The pseudocode for the combinatorial Benders' cut procedure is shown in Algorithm 1. In build_master_problem, the MP is built, which is the FBA problem with binary variables $\boldsymbol{a}$ that are linked to the directionality of $\mathbf{v}$. The relation between the flux variables $\mathbf{v}$ and the binary variables $\boldsymbol{a}$ can be expressed by indicator constraints as in Problem (47), by big-M constraints, or by indicator constraints and big-M constraints simultaneously. The big-M constant corresponds to the maximal absolute value of lower bounds $\mathbf{l}$ and upper bounds $\mathbf{u}$. The set of minimal infeasible subsystems is computed in compute_mis, which identifies at most m subsystems. The MIS search is explained in detail in Section 3.5. The dual problem of the infeasible subproblem is built with the Dualization package [13]. If we find a minimal infeasible subsystem $\mathcal{C}$, a combinatorial Benders' cut is added to the master problem. If there exists no MIS, the subproblem is feasible and thus the solution to the master problem is thermodynamically feasible. This process is repeated until a solution is found that is feasible in the master problem and the subproblem and therefore also for ll-FBA (indicator). In build_sub_problem, the subproblem is built based on the thermodynamically feasible solution of the master problem to obtain the corresponding values for $\boldsymbol{\Delta\mu}$ and $\boldsymbol{\mu}$.

Algorithm 1 solving ll-FBA with the combinatorial Benders' approach

Require: stoichiometric matrix $\mathbf{S}$, lower bound on fluxes $\mathbf{l}$, upper bound on fluxes $\mathbf{u}$, allowed number of cuts per iteration m, indices of internal reactions $\mathcal{I}$

MP $\leftarrow$ build_master_problem$(\mathbf{S}, \mathbf{l}, \mathbf{u}, \mathcal{I})$
$\mathbf{v}, \mathbf{a} \leftarrow$ optimize(MP)
$\mathcal{C} \leftarrow$ compute_mis$(\mathbf{S}_\mathcal{I}, \boldsymbol{a}, m)$ ▷ set of minimal infeasible subsystems
while $\mathcal{C} \neq \emptyset$ **do**
    add_cut$(\text{MP}, \boldsymbol{a}, \mathcal{C})$ ▷ combinatorial Benders' cut
    $\mathbf{v}, \boldsymbol{a} \leftarrow$ optimize(MP)
    $\mathcal{C} \leftarrow$ compute_mis$(\mathbf{S}_\mathcal{I}, \boldsymbol{a}, m)$
end while
SP $\leftarrow$ build_sub_problem$(\mathbf{S}_\mathcal{I}, \mathcal{I}, \boldsymbol{a})$
$\boldsymbol{\Delta\mu}, \boldsymbol{\mu} \leftarrow$ optimize(SP)
return $(\mathbf{v}, \boldsymbol{a}, \boldsymbol{\Delta\mu}, \boldsymbol{\mu})$ ▷ loopless flux distribution

Going back to the metabolic network in Figure 11 with the objective function $f = \max \mathbf{1}^\intercal \mathbf{v}$, the optimal solution of the master problem is $\mathbf{v}^* = (20, 30, 30, -20, 20, 10, 10)$ and $\boldsymbol{a} = (1, 1, 0, 1, 1)$. The solution is not thermodynamically feasible, and a minimal infeasible subsystem is $\mathcal{C} = \{3, 4, 5\}$, which corresponds to the cycle $r_4 = -10, r_6 = 10, r_7 = 10$. The corresponding combinatorial Benders' cut is:

$$a_3 + 1 - a_4 + 1 - a_5 \geq 1$$

As in the first iteration of the no-good cut approach, after adding the combinatorial Benders' cut, the optimal solution is $\mathbf{v}^* = (20, 30, 30, -10, 20, 0, 0)$. However, with combinatorial Benders' cuts, we need 2 iterations to obtain the ll-FBA solution, whereas with no-good cuts we need 5 iterations (see Figure 11).

MIS Search

If a solution to the master problem is not loopless, the subproblem is infeasible. We use the infeasible subproblem and its corresponding dual problem to generate a minimal infeasible subsystem.

First, the primal problem (Problem (49)) is rewritten in inequality form. To have the decision variable $\boldsymbol{\mu}$ on the left-hand side of $\leq$, the Constraint (49c) is multiplied by -1:

$$\Delta\mu_i \geq \epsilon \implies -\Delta\mu_i \leq -\epsilon$$

We also substitute $\boldsymbol{\Delta\mu}$ with $\mathbf{S}_{\mathcal{I}}^{\intercal}\boldsymbol{\mu}$ and obtain:

$$\max_{\mathbf{x}} \quad 0 \tag{50a}$$
$$\text{s.t.} \quad \tilde{\mathbf{A}}\boldsymbol{\mu} \leq \tilde{\mathbf{b}} \tag{50b}$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} [\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 1 \\ -[\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 0 \end{bmatrix} \quad \tilde{\mathbf{b}} = \begin{bmatrix} -\epsilon^{|\mathcal{C}|} \end{bmatrix}$$

We compute minimal infeasible subsystems by modifying the dual problem as explained in Section 1.4.2. The objective function of the dual is moved to the constraint $-\tilde{b}^{\intercal}\lambda = -1$, and we minimize $\sum_i w_i\lambda_i$. By modifying $\mathbf{w}$, we potentially derive several minimal infeasible subsystems to one infeasible solution $\boldsymbol{a}$. Each solution corresponds to a minimal infeasible subsystem. The nonzero elements in λ correspond to a MIS with the set of reaction indices $\mathcal{C}$. We choose how many cuts k can be added per iteration relative to the model size. For any $i \in \{1, ..., k\}$, we set the i-th coefficient in the objective function to zero and set all other coefficients to 1. At the end of the MIS search, we filter the minimal infeasible subsystems found to have unique subsystems within each iteration.

3.6 Intersection Cuts

In order to apply intersection cuts to ll-FBA (Problem (40)), the problem has to be brought in the required form as explained in Section 1.4.3:

$$\max_{\mathbf{x}} \quad \mathbf{c}^{\intercal}\mathbf{x}$$
$$\text{s.t.} \quad \mathbf{x} \in S \cap P$$

where P is polyhedral and S is closed. We define $\mathbf{x} = (\mathbf{v}, \boldsymbol{\Delta\mu}, \boldsymbol{\mu})$, where $\mathbf{v} \in \mathbb{R}^n$, $\boldsymbol{\Delta\mu} \in \mathbb{R}^{\dim \mathcal{I}}$ and $\boldsymbol{\mu} \in \mathbb{R}^m$. The vector $\mathbf{c} \in \mathbb{R}^n$ has the same coefficients as the objective function in the FBA problem and a coefficient of 0 for the variables $\boldsymbol{\Delta\mu}, \boldsymbol{\mu}$. The polyhedral set P contains solutions that respect the steady-state constraints, the bound constraints on $\mathbf{v}$ and Constraint (40e):

$$P = \{(\mathbf{v}, \boldsymbol{\Delta\mu}, \boldsymbol{\mu}) \,|\, \mathbf{Sv} = \mathbf{0},\, \mathbf{l} \leq \mathbf{v} \leq \mathbf{u},\, \boldsymbol{\Delta\mu}^{\intercal} = \boldsymbol{\mu}^{\intercal}\mathbf{S}_{\mathcal{I}}\}$$

Constraint (40d) is captured in the set S by using the same approximation as in Problem (44) and Problem (45): for any internal reaction i the flux through the reaction v_i and $\Delta\mu_i$ are of opposite sign unless $v_i = 0$ in which case $\Delta\mu_i$ can be positive or negative but cannot be in the interval $(-\epsilon, \epsilon)$. The set S is closed as it is defined using inclusive linear inequalities.

$$S = \{(v, \Delta\mu) \,|\, (v_i \geq 0,\ \Delta\mu_i \leq -\epsilon) \vee (v_i \leq 0,\ \Delta\mu_i \geq \epsilon) \quad \forall i \in \mathcal{I}\}$$

The intersection of P and S is the feasible region of the ll-FBA problem.
As explained in Section 1.4.3, in order to get an intersection cut we require an S-free set C containing the relaxed solution $\tilde{\mathbf{x}}$. Let $\tilde{\mathbf{x}} = (\tilde{\mathbf{v}}, \tilde{\boldsymbol{\Delta\mu}}, \tilde{\boldsymbol{\mu}})$ be a relaxed solution which is in P but not in S. There must be at least one reaction i for which holds $\tilde{\Delta\mu_i}\tilde{v}_i > 0$ or $\tilde{\Delta\mu_i} \in (-\epsilon, \epsilon)$. One of the S-free sets below contains such a relaxed solution:

$$C_i^{(1)} = \{(\mathbf{v}, \boldsymbol{\Delta\mu}) \,|\, \Delta\mu_i \geq -\epsilon,\ v_i \geq 0\}$$
$$C_i^{(2)} = \{(\mathbf{v}, \boldsymbol{\Delta\mu}) \,|\, \Delta\mu_i \leq \epsilon,\ v_i \leq 0\}$$

If a point is not in S, it will be either in $C_i^{(1)}$ or $C_i^{(2)}$, as the complement of their union together with the boundary is exactly S. As $C_i^{(1)}$ and $C_i^{(2)}$ are disjoint apart from the boundary, we conclude that both are maximal S-free sets. Suppose that $\tilde{v}_i > 0$ and $\tilde{\Delta\mu_i} > -\epsilon$. Such a solution projected into space of $(v_i, \Delta\mu_i)$ is not in S but contained in $C_i^{(1)}$. A visualization for this case is shown in Figure 6. If $\tilde{v}_i < 0$ and $\tilde{\Delta\mu_i} < \epsilon$ the solution is contained in $C_i^{(2)}$.

The conic relaxation P' at a relaxed solution $\tilde{\mathbf{x}}$ is a cone with apex $\tilde{\mathbf{x}}$ the extreme rays are the intersections of the active constraints. P' can be written as a conic combination of extreme rays or as a polyhedron of the active constraints (see Equations (23) and (24)). The basic variables and nonbasic constraints defining a basis $\tilde{\mathbf{A}}$ for $\tilde{\mathbf{x}}$ can be extracted from an LP solver.

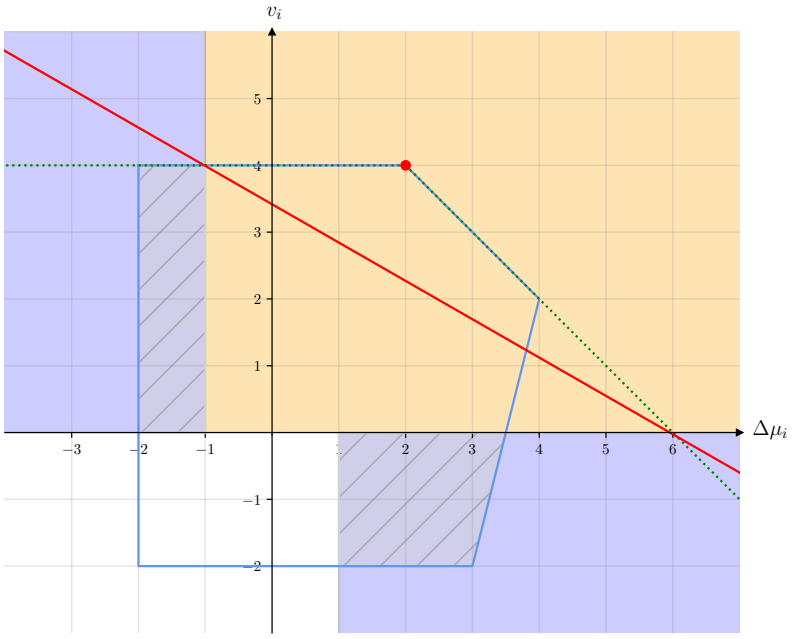


Figure 12: Visualization of an intersection cut for ll-FBA

We consider this simple model with constraints in 2D, with $\epsilon = 1$. We have the decision variables $v_i, \Delta\mu_i \in \mathbb{R}$, the polyhedral constraints P (blue), and the set S (purple area). The intersection of S and P is the true feasible region (gray-shaded area). Let the relaxed solution $(\tilde{\Delta\mu_i}, \tilde{v_i})$ (red point) be at $(2, 4)$. As $\tilde{\Delta\mu_i}$ and $\tilde{v_i}$ are both positive, $C_i^{(1)}$ is a maximal S-free set (yellow area). The conic relaxation at $(\tilde{\Delta\mu_i}, \tilde{v_i})$ and the S-free set $C_i^{(1)}$ intersect at the points $(-1, 4)$ and $(6, 0)$. The area between the line through the intersecting points and $(\tilde{\Delta\mu_i}, \tilde{v_i})$ is cut off.

Suppose reaction i in $\tilde{\mathbf{x}}$ has $\tilde{v_i} > 0$ and $\tilde{\Delta\mu_i} > -\epsilon$. It holds that $(\tilde{v_i}, \tilde{\Delta\mu_i}) \in C_i^{(1)}$ and $(\tilde{v_i}, \tilde{\Delta\mu_i}) \notin C_i^{(2)}$. To compute the intersection of the conic relaxation and $C_i^{(1)}$, we check for each extreme ray $\mathbf{r}^j = (-\tilde{\mathbf{A}}_{*,j}^{-1})$ generated by $\tilde{\mathbf{x}}$ whether it intersects the boundary of $C_i^{(1)}$. First, the intersection of the conic relaxation with the hyperplane $h_1 = \{\mathbf{x} \in \mathbb{R}^{n+m+|\mathcal{I}|} \mid v_i = 0\}$ is computed. We solve $(\tilde{\mathbf{x}} + \lambda_1 \mathbf{r}^j)_{k_1} = \mathbf{0}$ for λ_1, where k_1 corresponds to the index of v_i in $\tilde{\mathbf{x}}$.
Similarly, the intersection of each extreme ray generated by $\tilde{\mathbf{x}}$ with h_2 is computed, where $h_2 = \{\mathbf{x} \in \mathbb{R}^{n+m+|\mathcal{I}|} \mid \Delta\mu_i = -\epsilon\}$. We solve $(\tilde{\mathbf{x}} + \lambda_2 \mathbf{r}^j)_{k_2} = -\boldsymbol{\epsilon}$ for the step size λ_2, where k_2 is the index of $\Delta\mu_i$ in $\tilde{\mathbf{x}}$. An extreme ray intersects the boundary of $C_i^{(1)}$ if either λ_1 or λ_2 is finite and positive. If λ_1, λ_2 are both negative or if the ray does not intersect with either of the lines, there is no intersection with $C_i^{(1)}$ and the step size is set to ∞.
If reaction i in $\tilde{\mathbf{x}}$ has $v_i < 0$ and $\Delta\mu_i < \epsilon$, the intersection between the conic relaxation and $C_i^{(2)}$ is computed analogously. The equations to solve are $(\tilde{\mathbf{x}} + \lambda_1 \mathbf{r}^j)_{k_1} = \mathbf{0}$ and $(\tilde{\mathbf{x}} + \lambda_1 \mathbf{r}^j)_{k_2} = \boldsymbol{\epsilon}$.

However, after careful experimentation, we observe that deriving intersection cuts to solve ll-FBA is more challenging than expected. The relaxed problem is no longer full-rank, therefore the basis to $\tilde{\mathbf{x}}$ is not squared, which is required for the derivation of the intersection cut. The LP solver will internally transform the relaxed problem into a full-rank problem in a lower dimension. In order to have fewer decision variables in the original problem, we try to extract the basis of the relaxed problem of Problem (39), where we do not need the $\boldsymbol{\mu}$ variables. In our case, the $\boldsymbol{\Delta\mu}$ variables are removed during presolving. As our cut depends on the values of a pair of $\mathbf{v}$ and $\boldsymbol{\Delta\mu}$, we cannot use the basis of the transformed model. To show that a point lies on the boundary or on the interior of the convex hull of the feasible region of ll-FBA, which is the case for our nonbasic feasible solution, we solve a quadratic program for each internal reaction.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_i(\mathbf{x}) = ||\mathbf{x} - \mathbf{v}||^2 \\ \text{s.t.} \quad & x_i \in \mathcal{D}_i \end{aligned}$$

where $\mathbf{v}$ is the value of the solution of the relaxed problem and $\mathcal{D}_i$ is the convex-hull formulation of the internal reaction i (see Section 3.7). If $f_i(x) = 0$, $\mathbf{v}$ is in the convex hull or in the interior of the convex hull. In that case, we cannot derive an intersection cut and relaxed solution is not a basic feasible solution in the original variable space.
To obtain a basic feasible solution with a full-rank basis, we try to solve two auxiliary LPs such that their feasible region is defined by the following two polyhedral sets:

$$P_1 = \{\mathbf{v} \,|\, \mathbf{S}\mathbf{v} = \mathbf{0},\, \mathbf{l} \leq \mathbf{v} \leq \mathbf{u}\}$$

$$P_2 = \{(\boldsymbol{\Delta\mu}, \boldsymbol{\mu}) \,|\, \boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal \mathbf{S}_{\mathcal{I}}\}$$

We solve the FBA problem as auxiliary LP, to obtain a basic feasible solution in P_1. The second auxiliary with P_2 as its feasible region is solved to feasibility, which assigns the $\boldsymbol{\Delta\mu}$ variables. We can stack the solutions to the auxiliary programs P_1 and P_2 together. The corresponding matrix $\mathbf{B}$ is full-rank:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2 \end{bmatrix}$$

where the columns of $\mathbf{B}_1$ form a basis for a basic feasible solution in P_1, and the columns of $\mathbf{B}_2$ form a basis for a basic feasible solution in P_2. We can derive an intersection cut on the full-rank basis and the solution to the auxiliary LPs. We test

that the cut indeed removes the solution we derived the cut on. However, as the solution to the auxiliary LPs does not correspond to the relaxed solution, to which the cut is added, the relaxed solution does not change.
We see that in theory, we can use intersection cuts on the relaxed problem of ll-FBA. However, we require that a relaxed solution is a true basic feasible solution. More research on how to obtain a true basic feasible solution is needed, which is not part of this thesis. Therefore, we make no further experiments with this approach.

3.7 Disjunctive Programming

If we rewrite ll-FBA to have Boolean variables for each disjunct, we obtain the following program:

$$\max_{\mathbf{v},\boldsymbol{\Delta\mu},\boldsymbol{\mu}} \quad \mathbf{c}^\intercal \mathbf{v} \tag{51a}$$

$$\text{s.t.} \quad \mathbf{S}\mathbf{v} = \mathbf{0} \tag{51b}$$

$$\mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \tag{51c}$$

$$\boldsymbol{\Delta\mu}^\intercal = \boldsymbol{\mu}^\intercal \mathbf{S}_{\mathcal{I}} \tag{51d}$$

$$\begin{bmatrix} Y_i \\ v_i \geq 0 \\ \Delta\mu_i \leq -\epsilon \end{bmatrix} \vee \begin{bmatrix} Y_{i+|\mathcal{I}|} \\ v_i \leq 0 \\ \Delta\mu_i \geq \epsilon \end{bmatrix} \quad \forall i \in \mathcal{I} \tag{51e}$$

$$Y_i \vee Y_{i+|\mathcal{I}|} \quad \forall i \in \mathcal{I} \tag{51f}$$

where we have Boolean variables $Y_i \in \{true, false\}$ in addition to the continuous variables $\mathbf{v} \in \mathbb{R}^n, \boldsymbol{\Delta\mu} \in \mathbb{R}^{|\mathcal{I}|}, \boldsymbol{\mu} \in \mathbb{R}^m$.

The DisjunctiveProgramming package extends JuMP and enables formulating disjunctive programs and provides several MIP reformulations [31]. The package is used to model Problem (51) and to acquire the big-M reformulation and the hull reformulation (see Section 1.3). For both reformulations, the Boolean variable Y_i is modeled by a binary variable y_i. Constraint (51f) is then:

$$y_i + y_{i+|\mathcal{I}|} = 1 \quad \forall i \in \mathcal{I}$$

The big-M reformulation of Constraint (51e) is:

$$\begin{aligned} -v_i &\leq M(1 - y_i) \\ v_i &\leq M(1 - y_{i+|\mathcal{I}|}) \\ \Delta\mu_i &\leq -\epsilon + M(1 - y_i) \\ -\Delta\mu_i &\leq -\epsilon + M(1 - y_{i+|\mathcal{I}|}) \end{aligned}$$

which is identical to Problem (45) except that the big-M reformulation of Problem (51) uses two binary variables per disjunction instead of one. The tightness of the relaxed feasible solution of the big-M reformulation depends on the scalar M. The package contains the search for the minimum value of M by using interval arithmetic [1].
For the hull reformulation, two continuous variables are added for each decision variable in a disjunction. The variables corresponding to v_i are denoted by v_{i_1} and v_{i_2} and the variables corresponding to $\Delta\mu_i$ are $\Delta\mu_{i_1}$ and $\Delta\mu_{i_2}$. The hull reformulation of Constraint (51e) is:

$$\begin{aligned} v_i &= v_{i_1} + v_{i_2} \\ \Delta\mu_i &= \Delta\mu_{i_1} + \Delta\mu_{i_2} \\ -v_{i_1} &\leq 0 \\ v_{i_2} &\leq 0 \\ \Delta\mu_{i_1} &\leq -y_i \\ -\Delta\mu_{i_2} &\leq -y_{i+|\mathcal{I}|} \end{aligned}$$

We see that the hull reformulation needs more decision variables and constraints than the big-M reformulation. However, the feasible region of the hull reformulation is the convex hull of the disjunctions and therefore the tightest possible convex approximation.

3.8 Biological Models

3.8.1 BiGG

We use a subset of metabolic networks of the *biochemical, genetic, and genomic* (BiGG) database [18]. The models selected cover the different model sizes, the smallest being e_coli_core and the largest Recon3D. Table 1 lists the models including the number of metabolites and the number of reactions.

model	# metabolites	# reactions
e_coli_core	72	95
iAB_RBC_283	342	469
iIS312_Amastigote	606	519
iAF692	628	690
iSB619	655	743
iNF517	650	754
iHN637	698	785
iJB785	768	849
iNJ661	825	1025
iJN746	907	1054
iJR904	761	1075
iEK1008	998	1226
iCN900	885	1229
iYO844	990	1250
iND750	1059	1266
iMM904	1226	1577
iRC1080	1706	2191
iAF1260	1668	2382
iSDY_1059	1888	2539
STM_v1_0	1802	2545
iJO1366	1805	2583
iSbBS512_1146	1910	2591
iS_1188	1914	2619
iSFV_1184	1917	2621
iSF_1195	1917	2630
iSFxv_1172	1918	2638
iML1515	1877	2712
iZ_1308	1923	2721
iAPECO1_1312	1942	2735
iECB_1328	1951	2748
iETEC_1333	1962	2756
iYS1720	2436	3357
iMM1415	2775	3726
RECON1	2766	3741
iLB1027_lipid	2172	4456
Recon3D	5835	10600

Table 1: Model size of used BiGG models.

3.8.2 Yeast

We use a subset of metabolic networks of yeast models made available by [20]. The models selected are the smallest models and the model size is similar to the largest BiGG models selected.

model	**# metabolites**	**# reactions**
Hanseniaspora_uvarum	2464	3569
yHMPu5000035696_Hanseniaspora_singularis	2460	3534
yHMPu5000034963_Hanseniaspora_clermontiae	2464	3573
yHMPu5000035695_Hanseniaspora_pseudoguilliermondii	2476	3559
yHMPu5000035684_Kloeckera_hatyaiensis	2465	3582
Eremothecium_sinecaudum	2549	3471
yHMPu5000035659_Saturnispora_dispora	2598	3378
Tortispora_caseinolytica	2693	3597
Starmerella_bombicola_JCM9596	2695	3735
Eremothecium_gossypii	2556	3555
Ashbya_aceri	2553	3623

Table 2: Model size of used yeast models.

3.8.3 GECKO

When adding enzyme constraints to the model, the flux rate depends on the enzymes and the flux bounds $\mathbf{l}, \mathbf{u}$ are only required to restrict the direction of a reaction. We use COBREXA to build the enzyme models, and we generate the enzyme data randomly. The turnover numbers differentiate for the forward and backward direction of a reaction and therefore we split each reversible reaction into one forward and one backward reaction. At least one enzyme is mapped to each reaction including the turnover numbers for the reaction in the forward and the turnover number for the backward direction are defined. The turnover numbers are taken independently from a standard normal distribution. The protein concentrations have to be in the interval $[0, 1000]$. An enzyme is made up of one or more proteins. Each protein is randomly associated with either mass group A or B, and the product mass of each group is bounded by 0.5 $\frac{\text{mmol}}{\text{gDW}}$. For each protein we assign a product molar mass randomly from a Uniform distribution. The stoichiometric matrix including enzyme data $\mathbf{S}^{GECKO}$ has a row for each metabolite and for each protein in the network. The columns are the metabolic reactions in addition to an enzyme balance constraint linking each isozyme to its protein compound and the reaction it catalyzes.
The size of $\mathbf{S}^{GECKO}$ is therefore much larger than $\mathbf{S}$. As an example, the stoichiometric matrix of e_coli_core including enzyme data is of size $\mathbb{R}^{209\times334}$. The number of rows results from 72 metabolites and 137 proteins, and we have 334 columns, instead of 95.

Chapter 4

Results

In this section, we present the results of the computational experiments. The performance of solving the big-M reformulation of ll-FBA (Problem (40)) directly with SCIP is shown in Section 4.1. We experiment with blocking cycles (Section 4.2), decomposing the ll-FBA problem (Section 4.3) and the convex-hull formulation (Section 4.4).

The code[1] is written in Julia 1.9.0. We use JuMP [21] and MathOptInterface [19] to build the mathematical models. The MIP solver used in the experiments is SCIP [4]. The LP solver used is HiGHS [16]. We use COBREXA [9] to load the biological model data. The experiments were carried out on a 64-core compute node equipped with an Intel Xeon Gold 6338 2GHz CPU and 512GB RAM. The CPU memory was limited to 3000MB.

4.1 ll-FBA Variants

First, we look at the performance of solving the different ll-FBA variants by SCIP directly. We compare the running time and number of solved BiGG instances of ll-FBA (indicator) (Problem (44)), of ll-FBA (big-M) (Problem (45)) and of ll-FBA (nullspace) (Problem (38)).
The results are shown in Figure 13. We solve around 58% of the ll-FBA (big-M) instances, whereas around 31% of the ll-FBA (nullspace) instances are solved. We solve around 28% of the ll-FBA (indicator) instances. iSB619 is infeasible for ll-FBA (indicator) and is therefore excluded from the experiments with indicator constraints.

[1] available at
https://github.com/hannahtro/Loopless_Fluxes_with_Mixed_Integer_Optimization

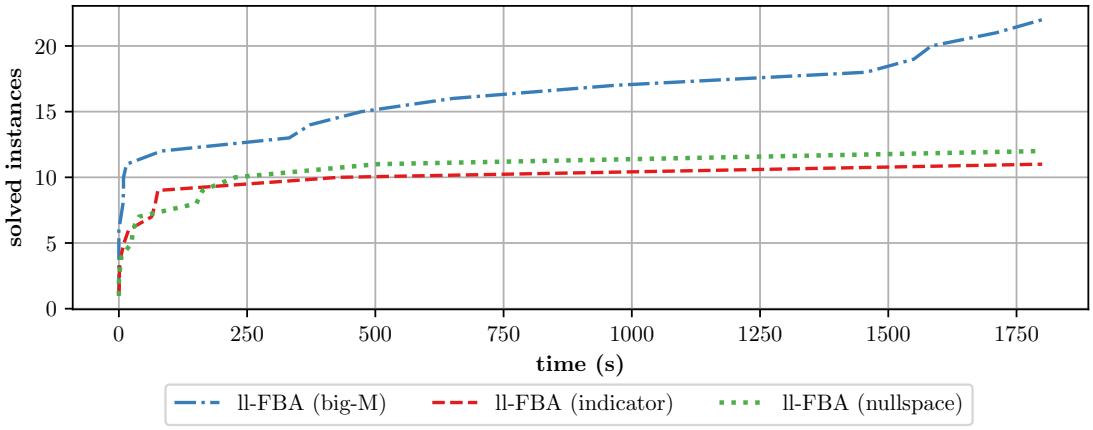


Figure 13: Performance of the different ll-FBA variants.

Comparing the number of optimally solved BiGG instances of different MIP formulations of ll-FBA within the time limit of 1800 seconds. The variants are solved by SCIP directly. We compare ll-FBA (big-M), ll-FBA (indicator) and ll-FBA (nullspace). The plot is based on Appendix Table 1 and Appendix Table 2.

As we solve twice as many ll-FBA (big-M) instances as ll-FBA (nullspace) instances, we use the ll-FBA (big-M) problem for the other experiments. Even though we also solve fewer ll-FBA (indicator) instances than ll-FBA (big-M) instances, we use the ll-FBA (indicator) for further experiments. Due to the explicit indicator constraints, the solving procedure is different compared to a mixed-integer linear program. With indicator constraints, fewer instances are solved, but some instances are solved much faster. For example, the ll-FBA (indicator) problems of models iMM1415 and iJN746 are solved in 71 and 10 seconds, whereas with the ll-FBA(big-M) formulations, we need 1550 and 332 seconds (see Appendix Table 1).

4.2 Blocking Cycles in ll-FBA

We experiment with blocking cycles in ll-FBA (big-M) as explained on a subset of BiGG instances (see Section 3.3). We then compare solving ll-FBA (big-M) directly to 8 different setups. We block any 10, 20, 50 and 100 cycles, and the 10, 20, 50 and 100 shortest cycles found prior to solving ll-FBA (big-M). We solve each setup with five different SCIP seeds for each model and set the time limit to 1800 minutes. The results are shown in Figure 14.

We see that the running time for the models e_coli_core, iCN900, iAF692 and iJR904 is only a few seconds and we see no impact of the different SCIP seeds. Model iML1515 cannot be solved within the given time limit by any solving strategy. For the other instances, we observe a small instance variability for ll-FBA (big-M) which shows that the difficulty is consistent across seeds and does not depend on the performance

variability of SCIP. The performance of the different solving strategies varies strongly. The instance variability of the setups differs significantly. Often, we see hardly any variability, while the running time for others varies up to 10 minutes, for example ll-FBA blocked 50 and ll-FBA blocked 100 on iSFV_1184, and for ll-FBA shortest blocked 20 and ll-FBA shortest blocked 100 on iMM904 . Some instances are solved faster by a blocking cycle strategy than solving ll-FBA (big-M) directly. However, the performance varies for different instances and seeds, and several strategies take much more time and some cannot solve within the time limit.

Figure 14 is not sufficient to conclude which setup outperforms the other. For such a statement we would require additional tests on more models. We also have to consider the time spent in the cycle search, which is not included in the figure (see Appendix Table 3). However, the results show that with cuts we can potentially improve the performance compared to solving ll-FBA (big-M) directly, which is tested in detail in the next section.

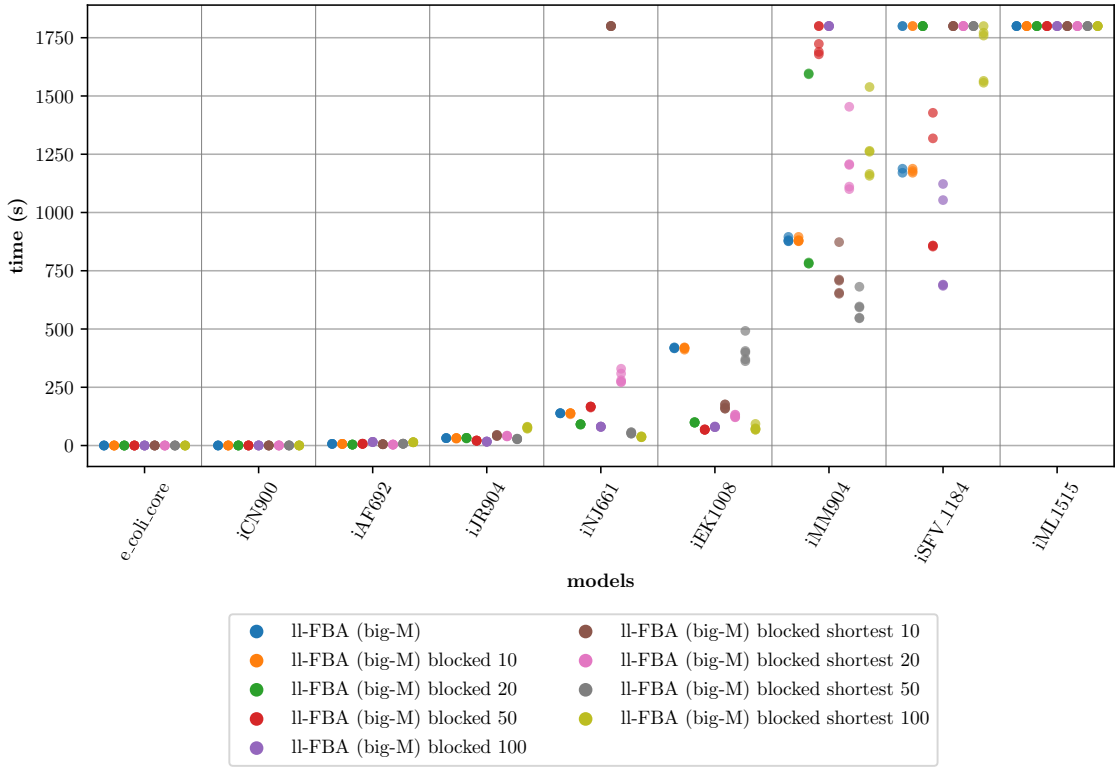


Figure 14: Performance of blocking cycle strategies

Comparing the running time of solving ll-FBA (big-M) directly and ll-FBA (big-M) with blocked cycles within the time limit of 1800 seconds. We experiment with blocking at most 10,20,50 and 100 (shortest) cycles. Each setup is tested with 5 different SCIP seeds.

4.3 Decomposition

Instead of solving the ll-FBA problem directly, we decompose the problem into a master problem, which is the relaxed ll-FBA problem, and add a cut if a relaxed solution does not satisfy the thermodynamic constraints.
We first compare solving ll-FBA (big-M) directly to solving the decomposition of ll-FBA (big-M) with no-good cuts (see Section 3.4). While we are able to solve around 60% by passing ll-FBA (big-M) directly to SCIP, we solve only 8% with no-good cuts. The ll-FBA (big-M) problem of model iAF692 is said to be infeasible with the no-good cut method. If we attempt to solve the instance with a higher precision, the instance cannot be solved within the time limit. Figure 15 shows the running time and the number of solved instances. With no-good cuts, we only block one assignment of binary variables $\boldsymbol{a}$ per iteration, therefore many cuts are required to obtain a loopless solution.

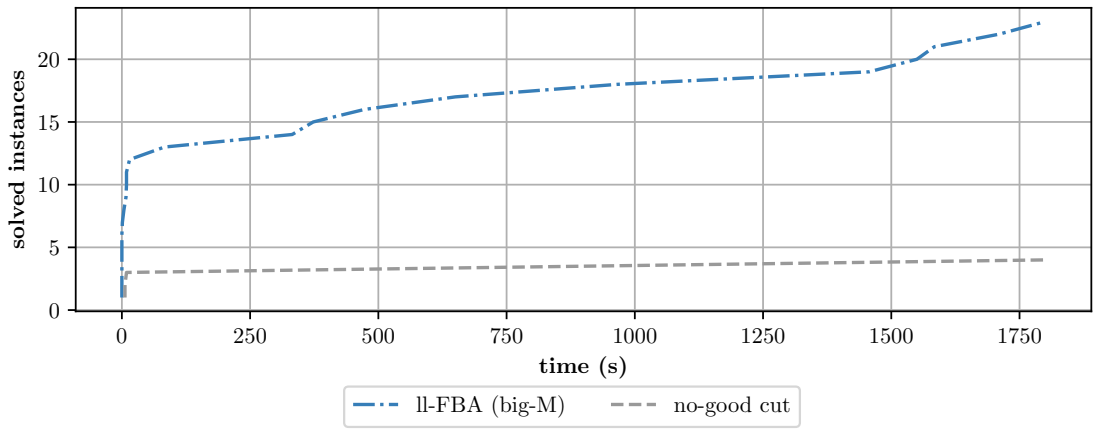


Figure 15: Performance of solving the decomposition with no-good cuts

Comparing the number of optimally solved BiGG instances solving ll-FBA (big-M) directly and solving the reformulation of ll-FBA (big-M) with no-good cuts within the time limit of 1800 seconds.

Next, we look at the comparison of the ll-FBA variants with the combinatorial Benders' approach. We experiment with three different formulations of the master problem (Problem (47)) and add one cut per iteration of Algorithm 1. We use:

1. the indicator formulation,
2. the big-M reformulation,
3. the indicator formulation and big-M reformulation.

Figure 16 shows the running time and the number of solved BiGG instances by the three combinatorial Benders' setups and by solving ll-FBA (big-M) and ll-FBA (indicator) with SCIP directly within a time limit of 1800 seconds. With the combinatorial Benders' approach, we are able to solve many more instances than by solving both ll-FBA variants directly. Combinatorial Benders' cuts are much stronger than no-good cuts. Instead of blocking the entire direction of binary variables $\boldsymbol{a}$ in the problem, we block one cycle per iteration.
Solving ll-FBA (indicator) directly leads to 28% optimally solved instances, whereas with combinatorial Benders' method with indicator constraints (CB (indicator)) we solve 84% of the instances to optimality. Solving ll-FBA (big-M) directly leads to 58% optimally solved instances, whereas with combinatorial Benders' with big-M constraints (CB (big-M)) we solve 92% of the instances to optimality.

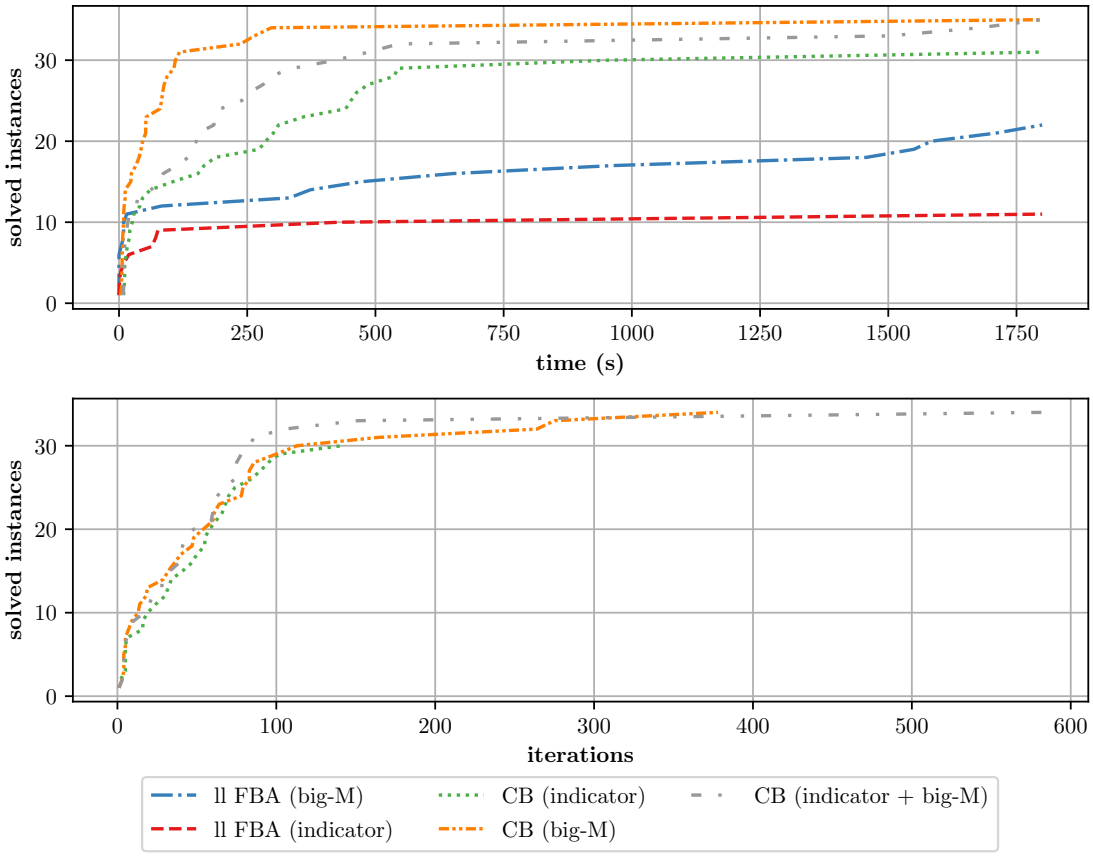


Figure 16: Performance of CB master problem variants

Comparing the number of optimally solved BiGG instances of ll-FBA (big-M), ll-FBA (indicator) directly and solving the combinatorial Benders' decomposition of ll-FBA within the time limit of 1800 seconds. We experiment with different formulations of the master problem: big-M reformulation, indicator constraints, and indicator and big-M constraints. Per iteration of the combinatorial Benders' approach, one minimal infeasible subsystem is computed.

However, Table 3 shows that for the combinatorial Benders' decomposition of ll-FBA (big-M), one instance is said to be infeasible, even though solving ll-FBA (big-M) returns an optimal solution. If we solve the instance with a tighter tolerance, we find an optimal solution in 264 seconds, whereas solving the ll-FBA (big-M) of iRC1080 directly takes 373 seconds.

		ll-FBA (indicator)		**CB (indicator)**		
time (s)	**# instances**	**# optimal**	**# time limit**	**# optimal**	**# time limit**	**# error**
0-10	10	6	3	9	0	1
10-600	5	3	2	3	2	0
600-1800	6	1	5	5	1	0
1800-Inf	15	0	15	13	2	0
		ll-FBA (big-M)		**CB (big-M)**		
time (s)	**# instances**	**# optimal**	**# time limit**	**# optimal**	**# time limit**	**# error**
0-10	10	10	0	10	0	0
10-600	5	5	0	4	0	1
600-1800	6	6	0	6	0	0
1800-Inf	15	0	15	13	2	0

Table 3: Comparing the termination status of solving ll-FBA (indicator) directly and solving ll-FBA (indicator) with the combinatorial Benders' decomposition within a time limit of 1800 seconds (above). Below we compare the termination status of solving ll-FBA (big-M) directly and solving ll-FBA (big-M) with the combinatorial Benders' decomposition. Per iteration of the combinatorial Benders' approach, one minimal infeasible subsystem is computed. The BiGG instances are divided depending on the running time of solving the corresponding ll-FBA variant with SCIP directly.

So far, we added one combinatorial Benders' cut per iteration of Algorithm 1. In Figure 17 we compare the performance of solving ll-FBA (indicator) directly with solving it with the combinatorial Benders' method by experimenting with the number of cuts added per iteration (see Section 3.5). As expected, using more cuts than one per iteration reduces the running time. Any combinatorial Benders' setup with multiple cuts per iteration solves faster than adding a single cut per iteration. For most setups, we see that more cuts lead to fewer iterations. However, too many cuts per iteration can lead to more iterations and a worse overall running time. For example, when blocking up to 20% of the number of reactions in the model per iteration, the number of iterations increases for some instances and the overall performance is reduced.

Table 4 shows that more cuts per iteration affect the termination status marginally. The errored instance corresponds to model iSB169 as solving the ll-FBA (indicator) problem directly terminated due to infeasibility.

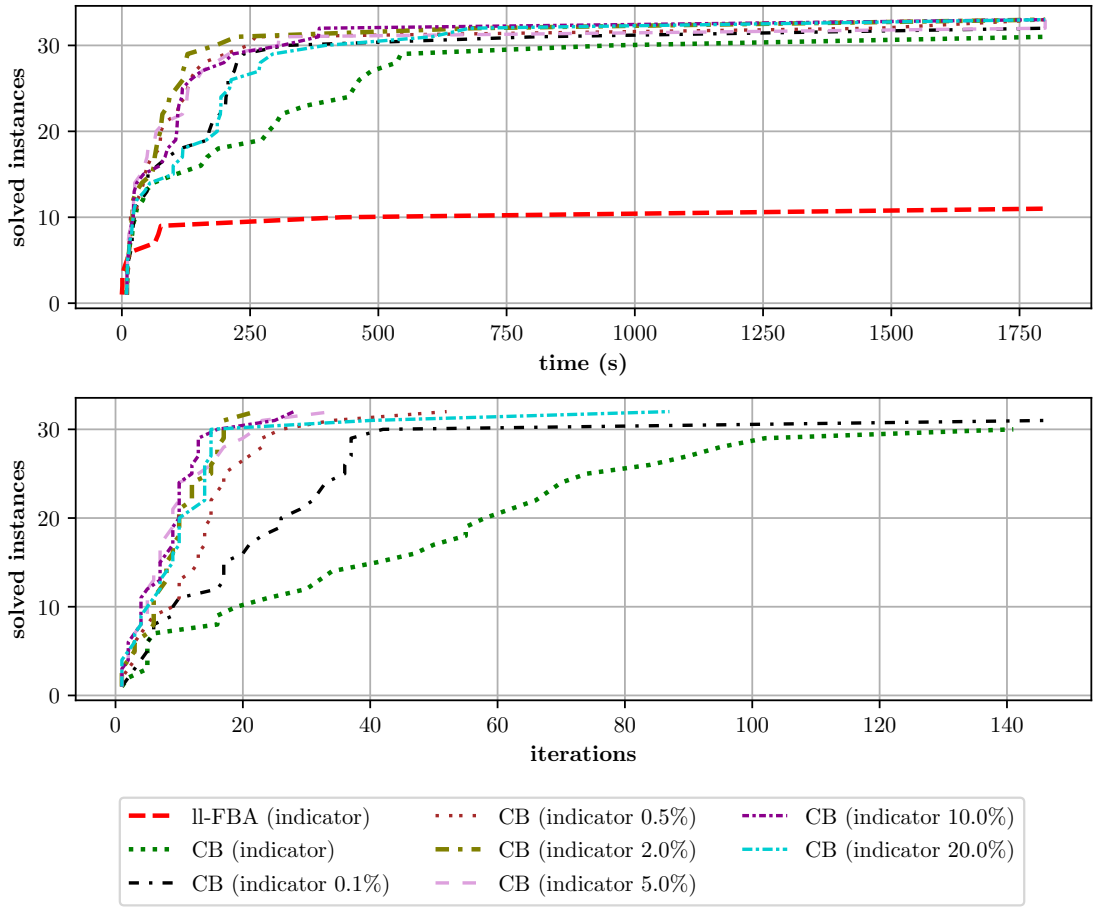


Figure 17: Performance of CB (indicator) with multiple cuts

Comparing the number of optimally solved BiGG instances of solving ll-FBA (indicator) directly and solving the combinatorial Benders' decomposition. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

	ll-FBA (indicator)		**CB (indicator)**		
MIS %	**# optimal**	**# time limit**	**# optimal**	**# time limit**	**# error**
0.1	10	25	31	4	1
0.5	10	25	32	3	1
1.0	10	25	32	3	1
2.0	10	25	32	3	1
5.0	10	25	32	3	1
10.0	10	25	32	3	1
20.0	10	25	31	4	1

Table 4: Comparing the termination status of solving ll-FBA (indicator) directly and solving ll-FBA (indicator) with the combinatorial Benders' decomposition within a time limit of 1800 seconds. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model. The BiGG instances are divided depending on the running time of solving ll-FBA (indicator) with SCIP directly.

In Figure 18, we see that with several cuts per iteration, the number of iterations is reduced while the time spent in the MIS search is larger. The average time spent in solving the master problem is smaller but overall much less than the time spent in the MIS search.

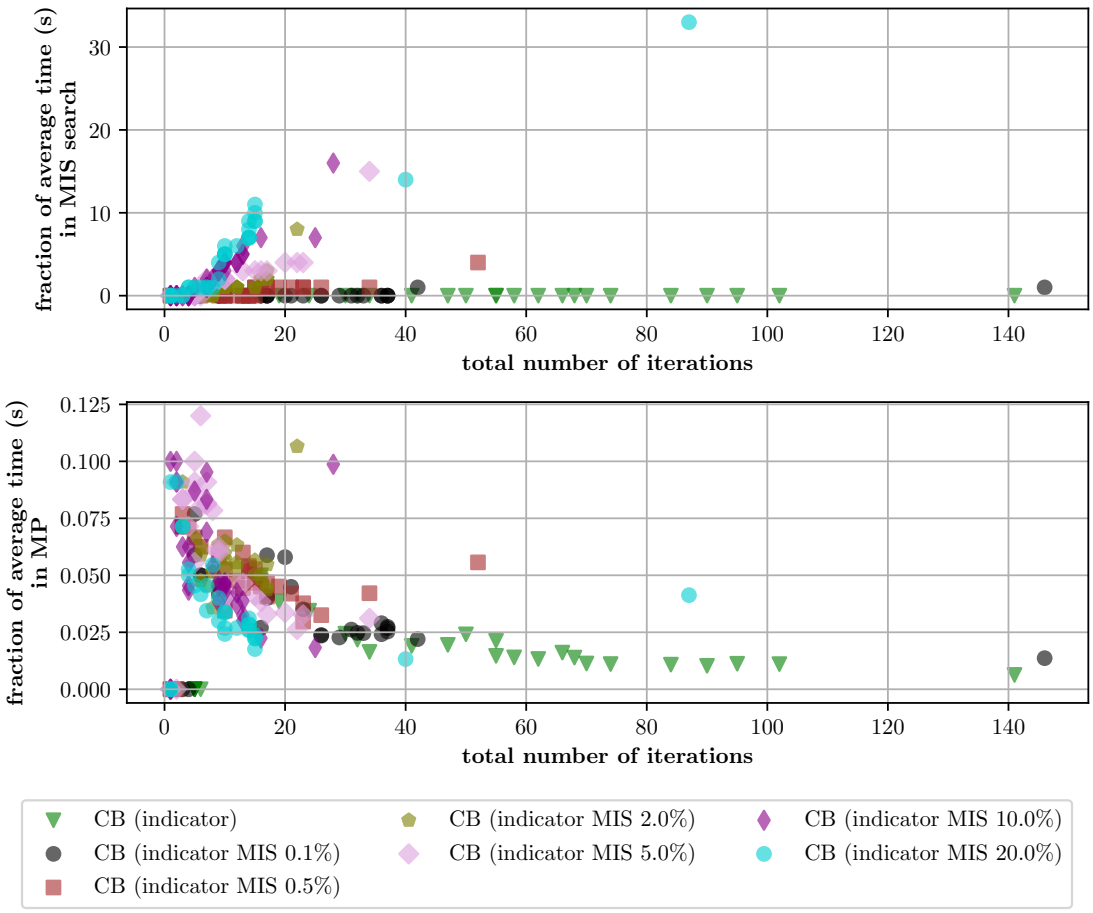


Figure 18: Time spent in master problem (indicator) and in MIS search

We compare the fraction of average time spent in the MIS search of the total running time to the total number of iterations solving ll-FBA (indicator) with the combinatorial Benders' approach on top. Below we compare the fraction of average time spent in the master problem of the total running time to the total number of iterations. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach depending on the instance size. MIS 0.5% means at most 0.5% of the number of reactions of the model. We compute the average with the geometric mean. We analyze BiGG instances where solving ll-FBA directly takes more than 15 seconds.

Next, we compare solving ll-FBA (big-M) directly to solving it with the combinatorial Benders' approach (see Section 3.5) by experimenting with the number of cuts added per iteration. With one cut per iteration, we already solve 92% of the BiGG instances to optimality and are faster than the combinatorial Benders' decomposition with indicator constraints. As with indicator constraints, with big-M constraints adding multiple cuts per iteration can speed up the solution process. However, adding MIS 2% or more cuts per iteration performs much worse than combinatorial Benders' with one cut per iteration and fewer iterations are solved within the time limit. With 0.5% MIS we see the best performance overall and require the least number of iterations per instance.

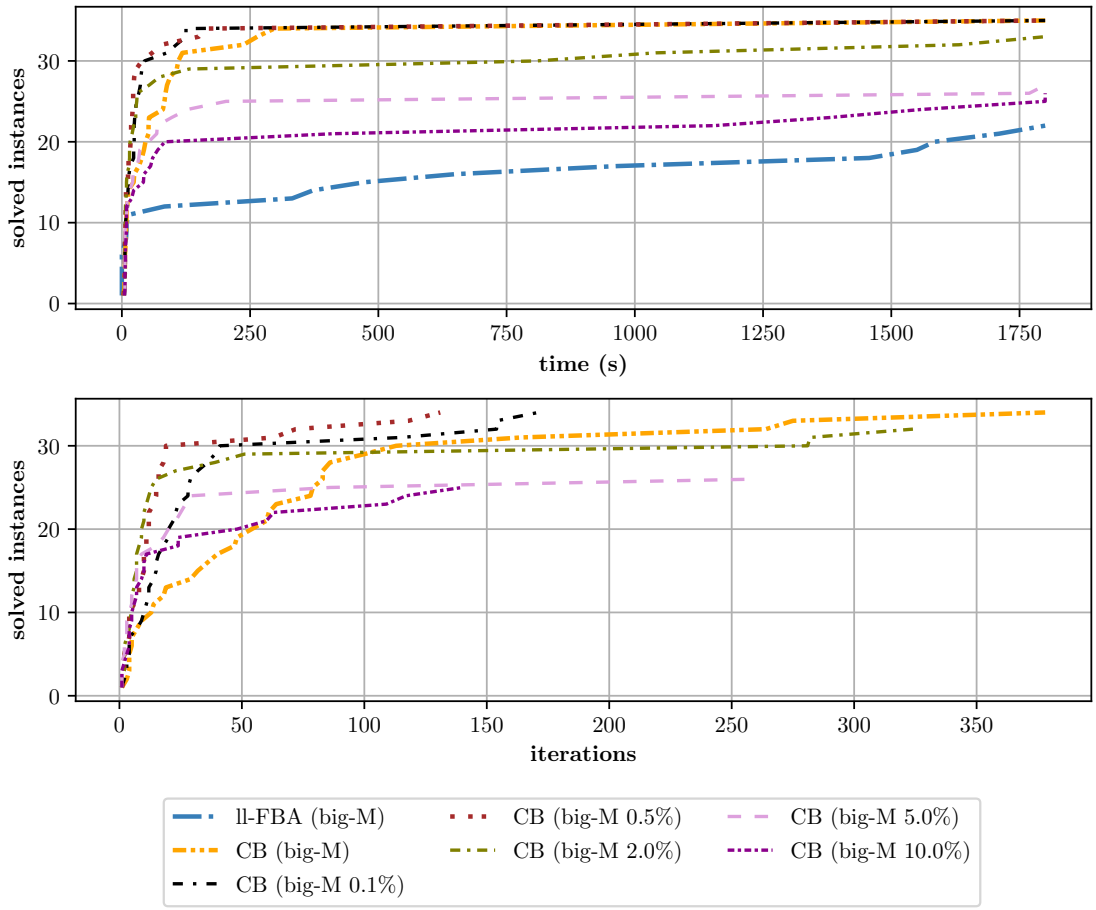


Figure 19: Performance of CB (big-M) with multiple cuts

Comparing the number of optimally solved BiGG instances of solving ll-FBA (big-M) directly and solving it with the combinatorial Benders' decomposition within a time limit of 1800 seconds. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

In Table 5 we see that with MIS 5.0% or an MIS 10 %, we solve significantly fewer instances due to hitting the time limit. The errored instance corresponds to the instance that is said to be infeasible (see Table 3). With a good choice of allowed cuts per iteration, we outperform the combinatorial Benders' method with indicator constraints. On the other hand, the choice of allowed cuts per iteration can increase the running time significantly in the combinatorial Benders' decomposition with big-M constraints.

	ll-FBA (big-M)		CB (big-M)		
MIS %	**# optimal**	**# time limit**	**# optimal**	**# time limit**	**# error**
0.1	21	15	32	3	1
0.5	21	15	32	3	1
1.0	21	15	32	3	1
2.0	21	15	31	4	1
5.0	21	15	26	9	1
10.0	21	15	24	11	1

Table 5: Comparing the termination status of solving ll-FBA (big-M) directly and solving ll-FBA (big-M) with the combinatorial Benders' method on the BiGG instances within a time limit of 1800 seconds. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

The comparison of the fraction of average time spent in the MIS search and in the master problem with big-M constraints in relation to the total number of iterations is less structured than the comparison with the indicator setting. Overall, the time spent in the MIS search and in solving the master problem is larger with more cuts, however, fewer iterations are to solve.

Instead of using JuMP to build the master problem, which builds a model in every iteration, we experiment with using a constraint handler in SCIP directly. Even though using a constraint handler should decrease the running time as the model is reused throughout the solving procedure, we do not see a better performance in preliminary experiments (see Appendix Table 6). We therefore continue with building the models in JuMP.

After experimenting with several formulations of the master problem and with the number of cuts per iteration of Algorithm 1, we test both solving strategies on the yeast models (see Section 3.8.2), which are larger than most BiGG models tested. In Table 6, we see that neither solving strategy performs well on the yeast instances. Only with one strategy we solve 1 out of 11 instances. On one yeast instance, we terminate due to an error on SCIP: the maximal depth level is exceeded even though it is set to 100,000.

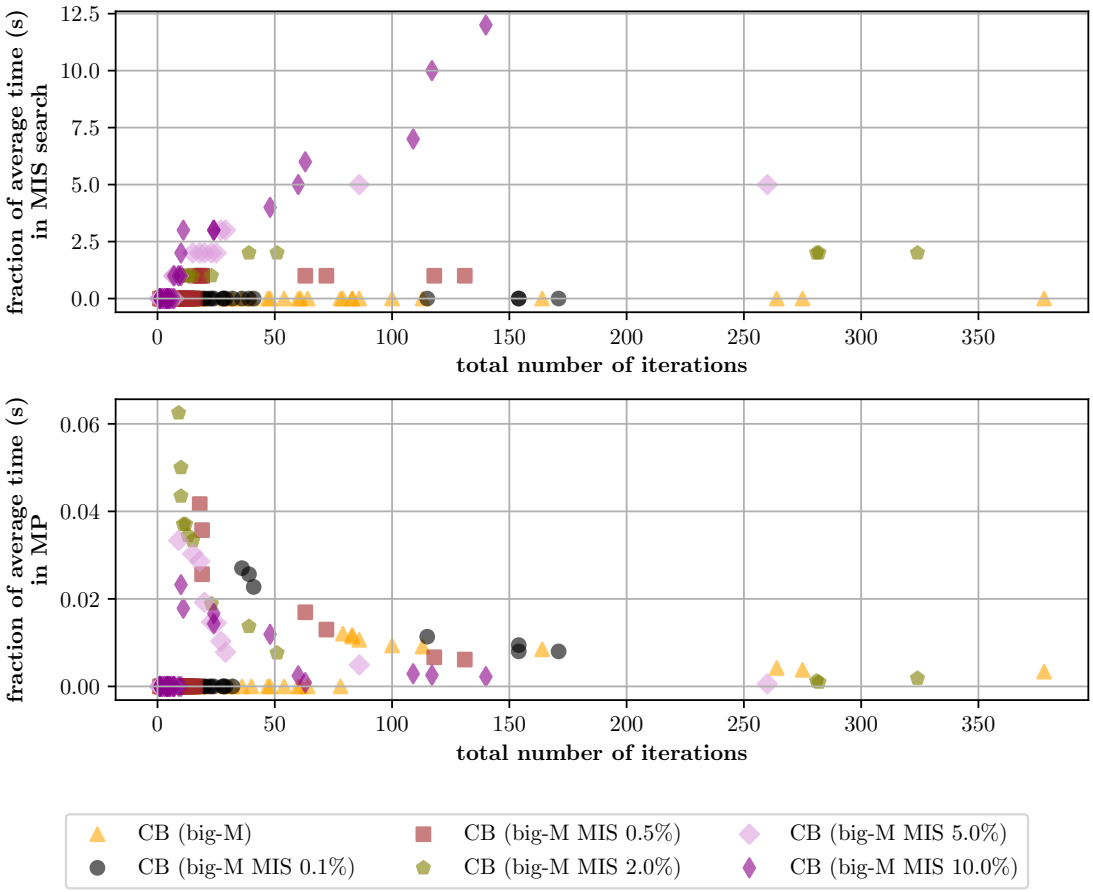


Figure 20: Time spent in master problem (big-M) and MIS search

We compare the fraction of average time spent in the MIS search of the total running time to the total number of iterations of solving ll-FBA (big-M) with the combinatorial Benders' decomposition on top. Below we compare the fraction of average time spent in the master problem of the total running time to the total number of iterations. We experiment with the number of cuts added per iteration of the combinatorial Benders' method depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model. We compute the average with the geometric mean. We analyze BiGG instances where solving ll-FBA directly takes more than 15 seconds.

Finally, we experiment with solving enzyme models with the combinatorial Benders' approach. We compare solving ll-FBA (big-M) directly to solving it with the combinatorial Benders' decomposition, as the big-M formulation showed the best performance on the other experiments. We compare blocking one cycle per iteration to blocking k cycles per iteration, where k is 0.5% of the number of reactions of the model. The results are shown in Figure 21. We see that both combinatorial Benders' variants solve similarly and outperform solving ll-FBA (big-M) directly. Both combinatorial Benders' variants require few iterations for most instances. As the reaction rate is limited by enzyme abundance, the relaxed solution contains already fewer loops. Therefore, less cuts are needed to obtain an optimal solution.

solving strategy	# optimal	# time limit	# error
ll-FBA (big-M)	0	11	0
CB (big-M)	0	11	0
CB (indicator)	0	11	0
CB (big-M MIS 0.1 %)	0	11	0
CB (big-M MIS 0.2 %)	0	11	0
CB (big-M MIS 0.5 %)	1	10	0
CB (big-M MIS 2.0 %)	0	11	0
CB (indicator MIS 0.1 %)	0	11	0
CB (indicator MIS 0.2 %)	0	11	0
CB (indicator MIS 0.5 %)	0	11	0
CB (indicator MIS 2.0 %)	0	10	1

Table 6: Comparing the termination status of solving yeast instances with ll-FBA (big-M) directly and solving ll-FBA (big-M) with the combinatorial Benders' method within a time limit of 1800 seconds. We experiment with different formulations of the master problem: indicator constraints, and big-M constraints. We compare adding a single cut per iteration of the combinatorial Benders' method denoted by CB and adding multiple cuts. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

However, looking at Table 7, we see that the combinatorial Benders' approach is not stable. We solve around 70% of instances to optimality but terminate the solution process in 18% of the instances not due to time limit but due to errors in the solving process. The enzyme models are numerically challenging. The majority of errors are due to an assertion to ensure that the objective value of the master problem in the first iteration is equal to the objective problem of the FBA problem within a tolerance of $1e-3$. In other instances, we cannot solve the subproblem due to numeric problems.

solving strategy	# optimal	# time limit	# error
ll-FBA (big-M)	16	74	0
CB (big-M)	63	5	22
CB (big-M MIS 0.5 %)	64	4	22

Table 7: Comparing the termination status of instances with enzyme constraints of solving ll-FBA (big-M) directly, and solving the problem with the combinatorial Benders' method within the time limit of 1800 seconds. We experiment with adding one cut per iteration of the combinatorial Benders' method and with adding at most k cuts per iteration, where k is 0.5% of the number of reactions in the model.

In our case, we are able to solve most enzyme instances within a few seconds. As the enzyme data is randomly generated, we do not make a claim on the performance on enzyme models in general. However, this shows the flexibility of the combinatorial Benders' approach, which can be extended by linear constraints easily.

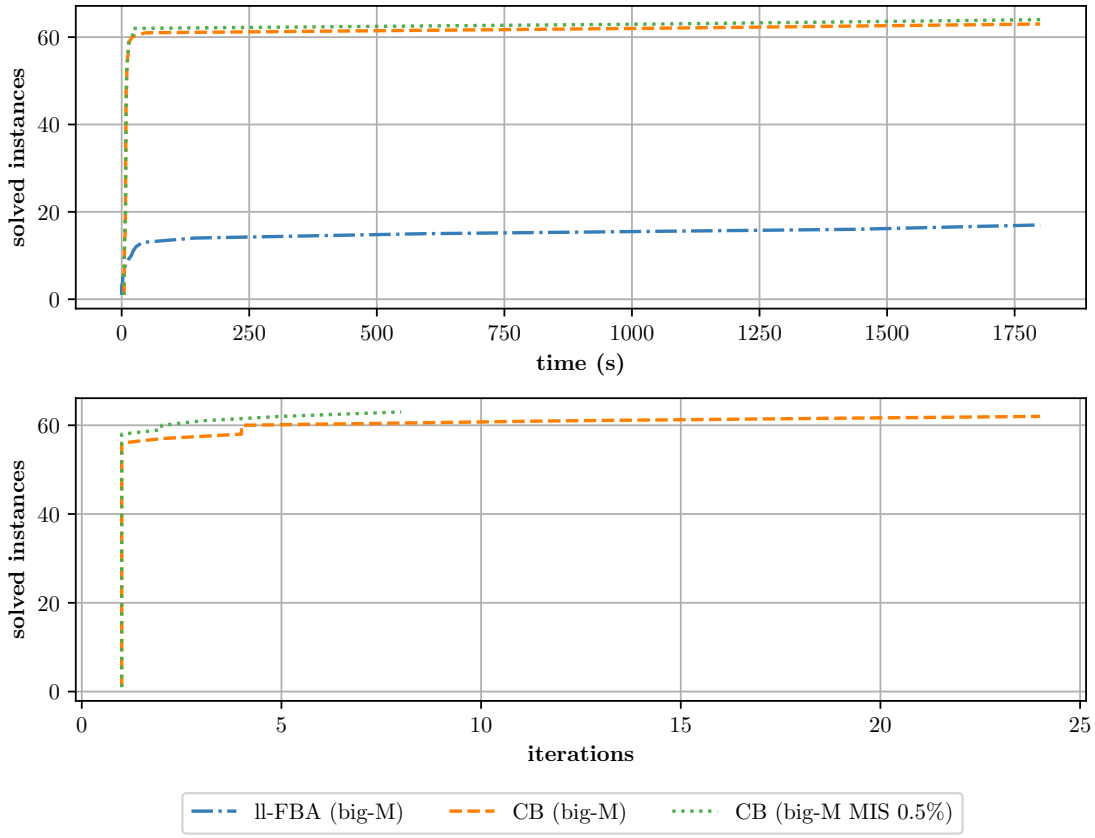


Figure 21: Performance of the combinatorial Benders' method on enzyme models

Comparing the number of optimally solved instances with enzyme constraints of solving ll-FBA (big-M) directly, and solving the problem with the combinatorial Benders' approach within the time limit of 1800 seconds. We experiment with adding a single cut per iteration of the combinatorial Benders' approach and with adding at most k cuts per iteration, where k is 0.5% of the number of reactions in the model.

4.4 Convex-Hull Formulation

We have seen that with cuts it is possible to reduce the running time of solving ll-FBA (indicator) and ll-FBA (big-M). Now we return to the disjunctive programming formulation of our problem. We use the DisjunctiveProgramming package to get the hull formulation and the big-M reformulation. We compare the performance of solving ll-FBA (big-M) directly, solving the big-M reformulation of the disjunctive program with optimizing the choice of M, the hull formulation of the disjunctive program and solving ll-FBA (big-M) with combinatorial Benders' cuts within a time limit of 1800 seconds. The difference between our big-M formulation is that in the package the choice of M is optimized.

Figure 22 shows that we solve the least instances with the convex-hull formulation. Even though the feasible region of the hull reformulation is the convex hull, the reformulation requires more constraints and decision variables, which leads to an

increased running time. With the big-M reformulation of the DisjunctiveProgramming package, we solve more instances than with our big-M formulation. The M constant determines how large the approximation of the feasible region is. Choosing the smallest M possible, we obtain a tighter approximation and more instances can be solved within the time limit. When we compare solving the MIP reformulations to solving ll-FBA (big-M) with combinatorial Benders' cuts, we see that with the latter we are able to solve most of the instances and are significantly faster.

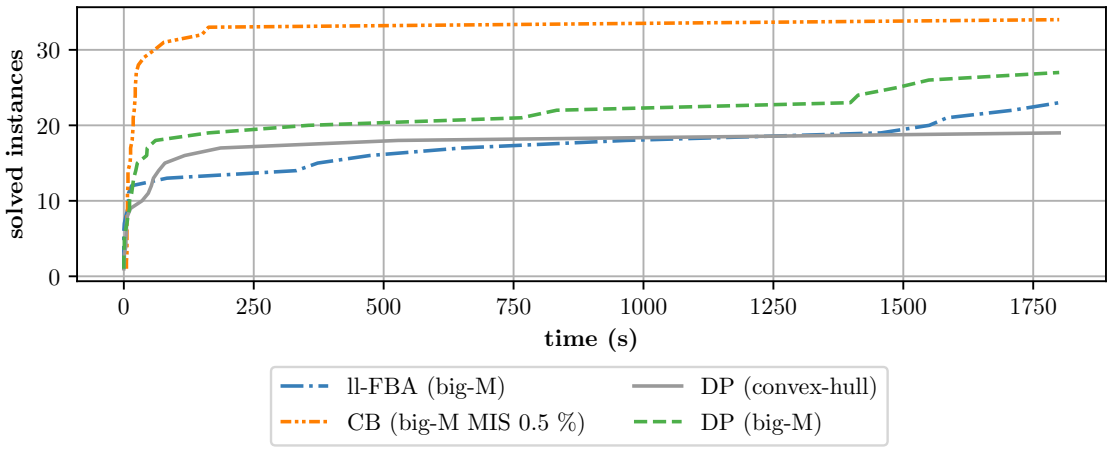


Figure 22: Performance of the big-M reformulation and the convex-hull formulation

Comparing the number of optimally solved BiGG instances of solving ll-FBA (big-M) directly, solving the convex-hull formulation (DP (convex-hull)), solving the big-M formulation of the DisjunctiveProgramming package with optimized M constant (DP (big-M)), and solving the problem with the combinatorial Benders' decomposition (CB (big-M MIS 0.5 %)) within the time limit of 1800 seconds.

Conclusion

The goal of this thesis was to compare the performance of different reformulations of loopless FBA (Problem (40)). We compared solving different reformulations of ll-FBA with SCIP directly including the big-M reformulation and the convex-hull formulation. With the big-M reformulation with an optimal constant M, we were able to solve most BiGG instances.

We then experimented with different solution methods in order to solve more instances than solving a reformulation of ll-FBA directly. We analyzed the performance if we detect cycles prior to solving the big-M reformulation of ll-FBA with blocking cycles with no-good cuts. We observed that the performance varies for different models and is often worse than solving the big-M reformulation of ll-FBA without blocking cycles.

We have seen how to apply intersection cuts to the ll-FBA problem in theory. As the relaxed solution is not a basic feasible solution in the original problem, we were not able to include intersection cuts in the experiments.

We experimented with decomposing the ll-FBA problem into a master and a subproblem, and compared solving it with no-good cuts and with combinatorial Benders' cuts. With no-good cuts, we were only able to solve a small subset of BiGG instances to optimality and the big-M reformulation of ll-FBA outperforms it. With combinatorial Benders' cuts, we solve more BiGG instances much faster than with the big-M reformulation of ll-FBA. However, the performance of the combinatorial Benders' approach depends strongly on the number of cuts added per iteration: too many cuts per iteration slow down the solution process significantly. On the yeast models, which are larger than most BiGG models we used, neither with the big-M reformulation nor with most combinatorial Benders' setups are we able to solve instances to optimality. The combinatorial Benders' approach is flexible and can be extended, as long as we can still decompose the problem into a master and a subproblem. If enzyme data is available, we can easily include enzyme constraints.

The different solution methods could be fine-tuned and optimized, which is briefly discussed in the following. For the blocking cycles approach, the number of blocked cycles should be linked to the instance size and we would have to test more instances. On the tested instances, the combinatorial Benders' approach is the most performant of the tested methods. When solving the big-M reformulation directly with optimized parameter M, we solve more instances than when using M fixed to the maximal absolute value of the upper and lower bounds on the fluxes. It would be interesting to incorporate the optimized big-M constant into the combinatorial Benders' approach. In several combinatorial Benders' setups, we see errors in the solution process especially due to numerical problems. As with the decomposition, the numerically tricky part is pushed to the subproblem which is a linear program, we could experiment with using exact LP to deal with the numerical instability [14].
We have seen that with the combinatorial Benders' approach, we can extend the model and include additional constraints. It would be interesting to use the combinatorial Benders' decomposition on the st-FBA problem, an extension to ll-FBA where the Gibbs free energy of a subset of reactions is constrained and energy-reducing cycles are valid solutions (see Section 2.2.5).

References

[1] A. Agarwal, S. Bhat, A. Gray, and I. E. Grossmann. Automating mathematical program transformations. In M. Carro and R. Peña, editors, *Practical Aspects of Declarative Languages*. Volume 5937. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-11503-5_12.

[2] E. Balas. *Disjunctive Programming*. Springer Publishing Company, Incorporated, 2018. DOI: 10.1007/978-3-030-00148-3.

[3] M. Basan, S. Hui, H. Okano, Z. Zhang, Y. Shen, J. R. Williamson, and T. Hwa. Overflow metabolism in escherichia coli results from efficient proteome allocation. *Nature*, 528(7580):99–104, Dec. 2015. DOI: 10.1038/nature15765.

[4] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The scip optimization suite 8.0, 2021. arXiv: 2112.08872 [math.OC].

[5] D. Bienstock, C. Chen, and G. Muñoz. Outer-product-free sets for polynomial optimization and oracle-based cuts. *Mathematical Programming*, 183, 2020. DOI: 10.1007/s10107-020-01484-3.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: https://doi.org/10.1017/CBO9780511804441.

[7] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54, 2006. DOI: 10.1287/opre.1060.0286. Publisher: INFORMS.

[8] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer Programming*, volume 271 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham, 2014. DOI: 10.1007/978-3-319-11008-0.

[9] Constraint-based reconstruction and exascale analysis. URL: https://github.com/LCSB-BioCore/COBREXA.jl (visited on 02/15/2024).

[10] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN: 0262033844.

[11] S. H. D. Dadush. Smoothed analysis of the simplex method. In *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021. DOI: 10.1017/9781108637435.

[12] A. A. Desouki, F. Jarre, G. Gelius-Dietrich, and M. J. Lercher. CycleFreeFlux: efficient removal of thermodynamically infeasible loops from flux distributions. *Bioinformatics*, 31, 2015. ISSN: 1367-4811, 1367-4803. DOI: 10.1093/bioinformatics/btv096.

[13] Dualization.jl. URL: https://github.com/jump-dev/Dualization.jl (visited on 02/15/2024).

[14] L. Eifler, J. Nicolas-Thouvenin, and A. Gleixner. Combining precision boosting with lp iterative refinement for exact linear optimization, 2023. arXiv: 2311.08037 [math.OC].

[15] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42, 1993. ISSN: 0166218X. DOI: 10.1016/0166-218X(93)90045-P.

[16] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10, 2018. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5.

[17] S. Huiberts. *How Large is the Shadow? Smoothed Analysis of the Simplex Method*. Master's thesis, Mathematical Sciences at Utrecht University, 2018. URL: https://studenttheses.uu.nl/handle/20.500.12932/28439.

[18] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson, and N. E. Lewis. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44, 2016. ISSN: 0305-1048. DOI: 10.1093/nar/gkv1049.

[19] B. Legat, O. Dowson, J. Dias Garcia, and M. Lubin. MathOptInterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689, 2021. DOI: 10.1287/ijoc.2021.1067.

[20] H. Lu, F. Li, L. Yuan, I. Domenzain, R. Yu, H. Wang, G. Li, Y. Chen, B. Ji, E. J. Kerkhoven, and J. Nielsen. Yeast metabolic innovations emerged via expanded metabolic network and gene positive selection. *Molecular Systems Biology*, 17, 2021. ISSN: 1744-4292. DOI: 10.15252/msb.202110427. Publisher: John Wiley & Sons, Ltd.

[21] M. Lubin, O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023. DOI: 10.1007/s12532-023-00239-3.

[22] MOSEK modeling cookbook. URL: https://docs.mosek.com/modeling-cookbook/index.html (visited on 04/22/2024).

[23] A. C. Müller and A. Bockmayr. Fast thermodynamically constrained flux variability analysis. *Bioinformatics*, 29, 2013. ISSN: 1367-4811, 1367-4803. DOI: 10.1093/bioinformatics/btt059.

[24] F. S. Musalem. *On cutting planes for mixed-integer nonlinear programming*. Dissertation, TU Berlin, 2021. DOI: 10.14279/depositonce-12180.

[25] E. Noor. Removing both internal and unrealistic energy-generating cycles in flux balance analysis, 2018. arXiv: 1803.04999 [q-bio.MN].

[26] E. Noor, N. E. Lewis, and R. Milo. A proof for loop-law constraints in stoichiometric metabolic networks. *BMC Systems Biology*, 2012. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-140.

[27] *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5.

[28] J. D. Orth, I. Thiele, and B. Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 2010. DOI: 10.1038/nbt.1614.

[29] B. Ø. Palsson. *Systems Biology: Constraint-based Reconstruction and Analysis*. Cambridge University Press, 2015. DOI: 10.1017/CBO9781139854610.

[30] A. Passi, J. D. Tibocha-Bonilla, M. Kumar, D. Tec-Campos, K. Zengler, and C. Zuniga. Genome-scale metabolic modeling enables in-depth understanding of big data. *Metabolites*, 12, 2021. DOI: 10.3390/metabo12010014.

[31] H. D. Perez, S. Joshi, and I. E. Grossmann. DisjunctiveProgramming.jl: generalized disjunctive programming models and algorithms for JuMP, 2023. arXiv: 2304.10492[cs].

[32] N. D. Price, I. Famili, D. A. Beard, and B. Ø. Palsson. Extreme pathways and kirchhoff's second law. *Biophysical Journal*, 2002. DOI: 10.1016/S0006-3495(02)75297-1.

[33] K. Raman. *An Introduction to Computational Systems Biology: Systems-Level Modelling of Cellular Networks*. Chapman and Hall/CRC, New York, 2021. DOI: 10.1201/9780429486951.

[34] K. Raman and N. Chandra. Flux balance analysis of biological systems: applications and challenges. *Briefings in Bioinformatics*, 10, 2009. DOI: 10.1093/bib/bbp011.

[35] A. C. Reimers. *Metabolic Networks, Thermodynamic Constraints, and Matroid Theory*. Dissertation, 2014. URL: http://dx.doi.org/10.17169/refubium-15314.

[36] B. J. Sánchez, C. Zhang, A. Nilsson, P.-J. Lahtvee, E. J. Kerkhoven, and J. Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. *Molecular Systems Biology*, 2017. DOI: 10.15252/msb.20167411.

[37] J. Schellenberger, N. E. Lewis, and B. Ø. Palsson. Elimination of thermodynamically infeasible loops in steady-state metabolic models. *Biophysical Journal*, 100(3), 2011. DOI: 10.1016/j.bpj.2010.12.3707.

[38] M. Terzer. *Large scale methods to enumerate extreme rays and elementary modes*. Doctoral Thesis, ETH Zurich, 2009. DOI: 10.3929/ethz-a-005945733.

[39] M. Turner, T. Koch, F. Serrano, and M. Winkler. Adaptive cut selection in mixed-integer linear programming. *Open Journal of Mathematical Optimization*, 2023. DOI: 10.5802/ojmo.25.

[40] *Understanding and Using Linear Programming*. Universitext. Springer, Berlin, Heidelberg, 2007. DOI: 10.1007/978-3-540-30717-4.

Appendix

	ll-FBA (big-M)			ll-FBA (nullspace)		
model	**termination**	**time**	**objective value**	**termination**	**time**	**objective value**
e_coli_core	OPTIMAL	0	0.874	OPTIMAL	0	0.874
iAB_RBC_283	OPTIMAL	0	2.936	OPTIMAL	2	2.936
iIS312_Amastigote	OPTIMAL	0	25.339	OPTIMAL	0	25.339
iAF692	OPTIMAL	0	0.0	TIME_LIMIT	1800	0.026
iSB619	OPTIMAL	0	0.0	TIME_LIMIT	1800	0.027
iNF517	OPTIMAL	9	0.043	OPTIMAL	39	0.043
iHN637	OPTIMAL	4	0.224	OPTIMAL	5	0.224
iJB785	OPTIMAL	15	0.054	OPTIMAL	25	0.0
iNJ661	TIME_LIMIT	1800	0.014	OPTIMAL	227	0.053
iJN746	OPTIMAL	332	1.4	TIME_LIMIT	1800	-
iJR904	OPTIMAL	9	0.0	OPTIMAL	163	0.922
iEK1008	OPTIMAL	473	0.0	OPTIMAL	503	0.058
iCN900	OPTIMAL	0	0.0	OPTIMAL	27	0.0
iYO844	TIME_LIMIT	1800	0.115	TIME_LIMIT	1800	0.0
iND750	OPTIMAL	8	0.0	OPTIMAL	151	0.0
iMM904	OPTIMAL	650	0.277	TIME_LIMIT	1800	0.0
iRC1080	OPTIMAL	373	0.0	TIME_LIMIT	1800	-
iAF1260	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iSDY_1059	OPTIMAL	1457	0.938	TIME_LIMIT	1800	0.0
STM_v1_0	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iJO1366	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iSbBS512_1146	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iS_1188	TIME_LIMIT	1800	0.832	TIME_LIMIT	1800	0.0
iSFV_1184	OPTIMAL	1710	0.894	TIME_LIMIT	1800	0.0
iSF_1195	OPTIMAL	1584	0.915	TIME_LIMIT	1800	0.0
iSFxv_1172	OPTIMAL	966	0.894	TIME_LIMIT	1800	0.0
iML1515	TIME_LIMIT	1800	0.861	TIME_LIMIT	1800	-
iZ_1308	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iAPECO1_1312	TIME_LIMIT	1800	0.934	TIME_LIMIT	1800	0.0
iECB_1328	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iETEC_1333	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	0.0
iYS1720	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iMM1415	OPTIMAL	1550	0.0	TIME_LIMIT	1800	-
RECON1	OPTIMAL	83	0.0	TIME_LIMIT	1800	-
iLB1027_lipid	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
Recon3D	TIME_LIMIT	1800	-	TIME_LIMIT	1800	-

Appendix Table 1: Results of solving ll-FBA (big-M) and ll-FBA (nullspace) directly with SCIP.

	ll-FBA (big-M)			ll-FBA (indicator)		
model	**termination**	**time**	**objective value**	**termination**	**time**	**objective value**
e_coli_core	OPTIMAL	0	0.874	OPTIMAL	0	0.874
iAB_RBC_283	OPTIMAL	0	2.936	OPTIMAL	1	2.936
iIS312_Amastigote	OPTIMAL	0	25.339	OPTIMAL	1	25.339
iAF692	OPTIMAL	0	0.0	TIME_LIMIT	1800	-
iSB619	OPTIMAL	0	0.0	INFEASIBLE	2	-
iNF517	OPTIMAL	9	0.043	OPTIMAL	19	0.043
iHN637	OPTIMAL	4	0.224	OPTIMAL	64	0.224
iJB785	OPTIMAL	15	0.054	OPTIMAL	429	0.054
iNJ661	TIME_LIMIT	1800	0.014	TIME_LIMIT	1800	0.0
iJN746	OPTIMAL	332	1.4	OPTIMAL	10	1.4
iJR904	OPTIMAL	9	0.0	TIME_LIMIT	1800	0.0
iEK1008	OPTIMAL	473	0.0	TIME_LIMIT	1800	0.0
iCN900	OPTIMAL	0	0.0	OPTIMAL	4	0.0
iYO844	TIME_LIMIT	1800	0.115	TIME_LIMIT	1800	0.118
iND750	OPTIMAL	8	0.0	TIME_LIMIT	1800	0.0
iMM904	OPTIMAL	650	0.277	TIME_LIMIT	1800	-
iRC1080	OPTIMAL	373	0.0	TIME_LIMIT	1800	-
iAF1260	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iSDY_1059	OPTIMAL	1457	0.938	TIME_LIMIT	1800	-
STM_v1_0	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iJO1366	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iSbBS512_1146	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iS_1188	TIME_LIMIT	1800	0.832	TIME_LIMIT	1800	0.0
iSFV_1184	OPTIMAL	1710	0.894	TIME_LIMIT	1800	-
iSF_1195	OPTIMAL	1584	0.915	TIME_LIMIT	1800	-
iSFxv_1172	OPTIMAL	966	0.894	TIME_LIMIT	1800	-
iML1515	TIME_LIMIT	1800	0.861	TIME_LIMIT	1800	-
iZ_1308	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iAPECO1_1312	TIME_LIMIT	1800	0.934	TIME_LIMIT	1800	0.0
iECB_1328	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iETEC_1333	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iYS1720	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
iMM1415	OPTIMAL	1550	0.0	OPTIMAL	71	0.0
RECON1	OPTIMAL	83	0.0	OPTIMAL	76	0.0
iLB1027_lipid	TIME_LIMIT	1800	0.0	TIME_LIMIT	1800	-
Recon3D	TIME_LIMIT	1800	-	TIME_LIMIT	1800	-

Appendix Table 2: Results of solving ll-FBA (big-M) and ll-FBA (indicator) directly with SCIP.

model	time (s)	cycles
e_coli_core	0.77	44
iCN900	50.47	191
iAF692	16.74	179
iJR904	42.58	489
iNJ661	38.08	510
iEK1008	58.04	547
iMM904	94.89	774
iSFV_1184	264.95	1100
iML1515	318.34	1237

Appendix Table 3: The running time and the number of detected cycles by using the CycleFreeFlux algorithm on a subset of BiGG models.

	ll-FBA (big-M)		no-good cuts	
model	termination	time	termination	time
e_coli_core	OPTIMAL	0	OPTIMAL	6
iAB_RBC_283	OPTIMAL	0	TIME_LIMIT	1800
iIS312_Amastigote	OPTIMAL	0	TIME_LIMIT	1800
iAF692	OPTIMAL	0	INFEASIBLE	-
iSB619	OPTIMAL	0	TIME_LIMIT	1800
iNF517	OPTIMAL	9	TIME_LIMIT	1800
iHN637	OPTIMAL	4	TIME_LIMIT	1800
iJB785	OPTIMAL	15	OPTIMAL	9
iNJ661	TIME_LIMIT	1800	TIME_LIMIT	1800
iJN746	OPTIMAL	332	TIME_LIMIT	1800
iJR904	OPTIMAL	9	TIME_LIMIT	1800
iEK1008	OPTIMAL	473	TIME_LIMIT	1800
iCN900	OPTIMAL	0	TIME_LIMIT	1800
iYO844	TIME_LIMIT	1800	TIME_LIMIT	1800
iND750	OPTIMAL	8	OPTIMAL	6
iMM904	OPTIMAL	650	TIME_LIMIT	1800
iRC1080	OPTIMAL	373	TIME_LIMIT	1800
iAF1260	TIME_LIMIT	1800	TIME_LIMIT	1800
iSDY_1059	OPTIMAL	1457	TIME_LIMIT	1800
STM_v1_0	TIME_LIMIT	1800	TIME_LIMIT	1800
iJO1366	TIME_LIMIT	1800	TIME_LIMIT	1800
iSbBS512_1146	TIME_LIMIT	1800	TIME_LIMIT	1800
iS_1188	TIME_LIMIT	1800	TIME_LIMIT	1800
iSFV_1184	OPTIMAL	1710	TIME_LIMIT	1800
iSF_1195	OPTIMAL	1584	TIME_LIMIT	1800
iSFxv_1172	OPTIMAL	966	TIME_LIMIT	1800
iML1515	TIME_LIMIT	1800	TIME_LIMIT	1800
iZ_1308	TIME_LIMIT	1800	TIME_LIMIT	1800
iAPECO1_1312	TIME_LIMIT	1800	TIME_LIMIT	1800
iECB_1328	TIME_LIMIT	1800	TIME_LIMIT	1800
iETEC_1333	TIME_LIMIT	1800	TIME_LIMIT	1800
iYS1720	TIME_LIMIT	1800	TIME_LIMIT	1800
iMM1415	OPTIMAL	1550	TIME_LIMIT	1800
RECON1	OPTIMAL	83	TIME_LIMIT	1800
iLB1027_lipid	TIME_LIMIT	1800	TIME_LIMIT	1800
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 4: Results of solving ll-FBA (big-M) directly and of solving the decomposition with no-good cuts.

	CB (indicator)		CB (big-M)		CB (indicator + big-M)	
model	**termination**	**time**	**termination**	**time**	**termination**	**time**
e_coli_core	OPTIMAL	9	OPTIMAL	5	OPTIMAL	8
iAB_RBC_283	OPTIMAL	11	OPTIMAL	5	OPTIMAL	9
iIS312_Amastigote	OPTIMAL	10	OPTIMAL	6	OPTIMAL	9
iAF692	OPTIMAL	16	OPTIMAL	7	OPTIMAL	9
iSB619	INFEASIBLE	-	OPTIMAL	6	INFEASIBLE	-
iNF517	OPTIMAL	29	OPTIMAL	12	OPTIMAL	21
iHN637	OPTIMAL	11	OPTIMAL	7	OPTIMAL	10
iJB785	OPTIMAL	14	OPTIMAL	7	OPTIMAL	11
iNJ661	OPTIMAL	61	OPTIMAL	9	OPTIMAL	33
iJN746	OPTIMAL	41	OPTIMAL	237	OPTIMAL	62
iJR904	OPTIMAL	21	OPTIMAL	12	OPTIMAL	37
iEK1008	OPTIMAL	20	OPTIMAL	9	OPTIMAL	17
iCN900	OPTIMAL	11	OPTIMAL	6	OPTIMAL	14
iYO844	OPTIMAL	26	OPTIMAL	8	OPTIMAL	15
iND750	OPTIMAL	46	OPTIMAL	10	OPTIMAL	81
iMM904	OPTIMAL	154	OPTIMAL	24	OPTIMAL	154
iRC1080	TIME_LIMIT	1800	INFEASIBLE	-	OPTIMAL	1502
iAF1260	OPTIMAL	166	OPTIMAL	39	OPTIMAL	198
iSDY_1059	OPTIMAL	273	OPTIMAL	46	OPTIMAL	141
STM_v1_0	OPTIMAL	105	OPTIMAL	23	OPTIMAL	86
iJO1366	OPTIMAL	286	OPTIMAL	52	OPTIMAL	135
iSbBS512_1146	OPTIMAL	360	OPTIMAL	118	OPTIMAL	239
iS_1188	OPTIMAL	311	OPTIMAL	52	OPTIMAL	185
iSFV_1184	OPTIMAL	442	OPTIMAL	85	OPTIMAL	254
iSF_1195	OPTIMAL	452	OPTIMAL	43	OPTIMAL	151
iSFxv_1172	OPTIMAL	301	OPTIMAL	53	OPTIMAL	186
iML1515	OPTIMAL	188	OPTIMAL	33	OPTIMAL	113
iZ_1308	OPTIMAL	536	OPTIMAL	83	OPTIMAL	294
iAPECO1_1312	OPTIMAL	954	OPTIMAL	107	OPTIMAL	422
iECB_1328	OPTIMAL	486	OPTIMAL	80	OPTIMAL	281
iETEC_1333	OPTIMAL	544	OPTIMAL	88	OPTIMAL	339
iYS1720	OPTIMAL	463	OPTIMAL	94	OPTIMAL	487
iMM1415	TIME_LIMIT	1800	OPTIMAL	110	OPTIMAL	545
RECON1	TIME_LIMIT	1800	OPTIMAL	297	TIME_LIMIT	1800
iLB1027_lipid	TIME_LIMIT	1800	TIME_LIMIT	1800	OPTIMAL	1678
Recon3D	TIME_LIMIT	1800	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 5: Results of solving the ll-FBA with combinatorial Benders' cuts for different formulations of the master problem. We add one cut per iteration of Algorithm 1.

	CB (big-M)		CH		CH MIS 5%	
model	**termination**	**time**	**termination**	**time**	**termination**	**time**
e_coli_core	OPTIMAL	5	OPTIMAL	6	OPTIMAL	5
iAF692	OPTIMAL	7	OPTIMAL	10	OPTIMAL	10
iNF517	OPTIMAL	12	OPTIMAL	28	OPTIMAL	19
iNJ661	OPTIMAL	9	OPTIMAL	24	OPTIMAL	24
iJR904	OPTIMAL	12	OPTIMAL	11	OPTIMAL	11
iEK1008	OPTIMAL	9	OPTIMAL	36	OPTIMAL	36
iCN900	OPTIMAL	6	OPTIMAL	5	OPTIMAL	6
iMM904	OPTIMAL	24	OPTIMAL	75	OPTIMAL	76
iAF1260	OPTIMAL	39	OPTIMAL	207	OPTIMAL	207
iSDY_1059	OPTIMAL	46	OPTIMAL	201	OPTIMAL	204
iJO1366	OPTIMAL	52	OPTIMAL	993	OPTIMAL	1039
iSbBS512_1146	OPTIMAL	118	TIME_LIMIT	1800	TIME_LIMIT	1800
iS_1188	OPTIMAL	52	OPTIMAL	225	OPTIMAL	226
iSFV_1184	OPTIMAL	85	OPTIMAL	1156	OPTIMAL	1155
iSF_1195	OPTIMAL	43	OPTIMAL	313	OPTIMAL	320
iML1515	OPTIMAL	33	TIME_LIMIT	1800	TIME_LIMIT	1800

Appendix Table 6: Results of solving ll-FBA with the combinatorial Benders' approach in JuMP (**CB (big-M)**) compared to integrating the Combinatorial Benders' decomposition in a constraint handler in SCIP directly (**CH**).